DDC

LEVEL II

BS

AD A100823

Best Available Copy

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

81 6 30 038

AFIT/GEO/EE/80D-2

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

(6) DIGITAL CONTROL SYSTEM FOR
AN ON-BOARD OXYGEN GENERATOR.

(9) Master's THESIS, (11) Dec 80

(14) AFIT/GEO/EE/80D-2 (10) Thomas C./Horch
Capt USAF (12) 88

Approved for public release; distribution unlimited.

AFIT/GEO/EE/80D-2

DIGITAL CONTROL SYSTEM FOR AN ON-BOARD OXYGEN GENERATOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Thomas C. Horch, B.S.
CAPT USAF
Graduate Electro-Optics

December 1980

Preface

A prototype On-Board Oxygen Generator system has been built and tested at the Air Force School of Aerospace Medicine. However, the oxygen generator must be controlled to generate a breathing product compatible with aircrew physiological requirements. This report describes a digital control system which is believed capable of controlling the generator under all anticipated conditions of use. At the time of this writing, the control system has met all requirements as specified by personnel at the School of Aerospace Medicine.

I would like to thank those people who contributed to the success of this thesis. They include the following: Major Borky, my thesis advisor; the entire staff of the Electrical Engineering Department technicians; Sharon Gabriel and my wife for typing this thesis. I would also like to thank my wife, Fran, for her continued support throughout my AFIT tour.

Thomas C. Horch

Contents

Contents (Cont'd)

List of Figures

Abstract

A prototype on-board oxygen generation system (OBOGS) is being tested by the USAF School of Aerospace Medicine (SAM). The OBOGS is a candidate to replace present liquid oxygen systems for aircrew consumption on manned aircraft. The OBOGS passes outside air through a molecular sieve to produce an oxygen enriched breathing product. Oxygen concentration of the OBOGS's output is controlled by a purge orifice valve. The SAM envisions using a digital system to control the OBOGS.

A digital control system for the OBOGS was developed and consists of a stepper motor, microprocessor, system sensors, support circuitry, and software. The control system software is a collection of instructions which allow the MPU to read data from sensors, to interpret that data, and to issue system hardware control signals. System software was fairly complex as methods were employed to compensate for the OBOGS's lengthy response time. This was accomplished by using a segmented table. If motor drive is anticipated to be time-consuming, a software routine is used to preposition the motor to a predetermined location within the segmented table. This position is updated when more accurate information is available.

A prototype system was constructed and tested in the laboratory. The control system successfully controlled the stepper motor.

DIGITAL CONTROL SYSTEM FOR
AN ON-BOARD OXYGEN GENERATOR

I. Introduction

Background

A molecular sieve oxygen generator (MSOG) is being developed by the USAF School of Aerospace Medicine (SAM). The MSOG will be utilized as an on-board oxygen generation system (OBOGS) to replace liquid oxygen (LOX) breathing systems in manned aircraft (Ref 11:1). Although the LOX system has historically been reliable, LOX is expensive and difficulties in handling LOX increase aircraft turn-around time. An OBOGS used filtered atmospheric air and produces an oxygen enriched product for aircrew breathing (Ref 11:1). Therefore, the OBOGS will eliminate costly liquid oxygen and the handling problems associated with LOX systems. In addition, the OBOGS filters most contaminants, including chemical warfare agents, which makes the OBOGS an extremely attractive alternative to LOX systems (Ref 11:1).

To be successful, the OBOGS must generate a breathing product compatible with aircrew physiological requirements. Human oxygen requirements are well known and are a function of atmospheric pressure altitude. At sea level, normal air with 20 percent oxygen at 1 atmosphere pressure is sufficient for aircrew consumption. As cabin altitude increases to 32,000 feet, oxygen concentration requirements increase nonlinearly to 100 percent oxygen (unpressurized).

Above 32,000 feet, 100 percent oxygen is required at pressures above 1 atmosphere.

The OBOGS produces an oxygen enriched, pressurized product which can be regulated with a conventional aircraft pressure regulator. Therefore, the physiological breathing gas pressure requirements of the OBOGS are controllable with standard pressure regulators. Oxygen concentration of the OBOGS's output can be controlled with a purge orifice valve (Ref 11:4). Experimental results indicate that the OBOGS's pressurized oxygen concentration output can be controlled from 20 percent to 95 percent (Ref 11:4). The aerospace medical community at Brooks Air Force Base is currently trying to verify that a 95 percent oxygen concentration is adequate in lieu of 100 percent oxygen (Ref 4).

Consequently, success of the OBOGS depends upon the ability to accurately position the OBOGS's purge orifice valve. However, positioning the valve is not a simple linear function of cabin altitude. Instead, oxygen concentration of the OBOGS's output is dependent upon the volume of breathing gas required for crew consumption (Ref 11:4). This volume is a function of the number of crew members and their individual requirements. Individual requirements vary with health, physical activity and state of excitement. These considerations must be used to formulate a control strategy.

An optimum method of OBOGS control is to determine the oxygen concentration requirements as a function of cabin altitude and then measure the actual oxygen concentration of the OBOGS's output. Next, an electronic system could compare these values and adjust the purge orifice valve accordingly (Ref 7). This method formed the basis in this thesis effort for constructing an OBOGS control system.

Problem Statement

The purpose of this thesis is to determine if a digital electronic control system is viable for OBOGS (Ref 7). The major components of the control system consist of a stepper motor to position the purge orifice valve, an oxygen sensor, an altitude transducer, and a microprocessor unit (MPU) (Ref 7). The MPU's inputs, oxygen concentration and cabin altitude, are used to calculate the proper commands for the stepper motor.

It should be noted that a solid-state oxygen sensor is currently being developed for the OBOGS. Response of this sensor is expected to be very non-linear, and its time response may be 10 seconds or more (Ref 4). Therefore, the control system must be designed to compensate for the oxygen sensor's performance. Even though exact performance characteristics of the oxygen sensor are presently unknown, it is believed that maximum flexibility can be incorporated in the control system by using the following system concept.

System Concept

The proposed OBOGS digital control system is shown in Figure 1. The MPU is programmed to perform a sequence of steps. The programmed steps are stored in an erasable programmable read-only memory (EPROM) integrated circuit. Power-on reset causes the MPU to fetch the instruction stored at address 000, and the remaining instructions are then executed sequentially. The MPU is programmed to accept an input from each of the sensors via an analog to digital (A/D) converter

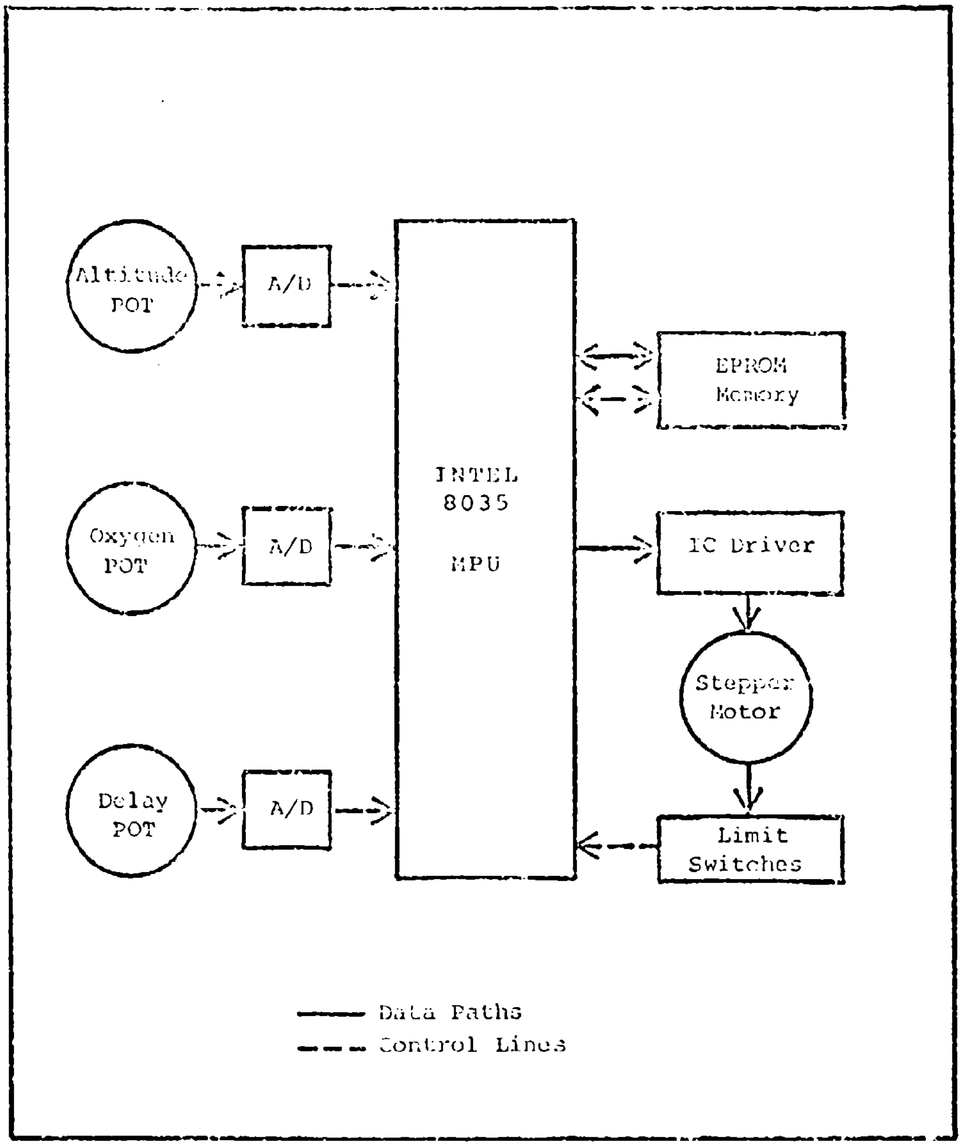


Figure 1. OBOGS's Control System

and to store these values in internal MPU registers. The altitude transducer input, simulated in the prototype by a potentiometer, is used as an entering argument in a software table to look-up the required physiological oxygen concentration. Similarly, the oxygen sensor input, also simulated by a potentiometer, is used to look-up the actual oxygen concentration in a table obtained from calibration measurements of the oxygen sensor's response. The delay potentiometer setting is also read by the MPU. These values are used as follows.

The MPU uses the derived table values to calculate the appropriate drive for the stepper motor. The direction of rotation is determined by comparing the desired oxygen concentration with the actual oxygen concentration. The outcome of this comparison is used to establish the rotation input signal for the stepper motor drive circuit. At this point, a program could have been written to increment the stepper motor. The program could then jump back to the beginning and form a continuous loop. However, such a program would not compensate for the OBOGS's response time.

The following items are incorporated to compensate for the slow system response time. The delay potentiometer is used to limit the MPU's rate of issuing step commands to the motor. This potentiometer is preset to a value which yields a controller cycle time equal to the OBOGS's response time. The OBOGS's response time is the time required for oxygen to travel through the OBOGS's plumbing and to be measured by the oxygen sensor after the purge orifice valve has been repositioned. The delay potentiometer is necessary to prevent the motor from oscillating about a desired position. However, the delay potentiometer can present a severe shortcoming. If the system response time is long (it is expected to be over 10 seconds), and if a rapid change in oxygen concentration requirement occurs (for example, rapid decompression), the time required to drive the motor to the new position could be beyond the aviator's consciousness limit. To solve this dilemma, a series of steps are incorporated in the program.

This thesis describes an active algorithm involving coarse and fine adjustments to reduce the control system's response time. This concept involves segmenting the oxygen demand schedule in 5 percent steps as shown in Figure 2. Note that the oxygen schedule for this prototype system uses a piecewise linear model and does not include upper or lower bounds. Each of the 16 steps has a corresponding register reserved in the MPU. These registers are called segmented registers and contain numbers called segmented values. Another register, the motor position counter, will contain a number that corresponds to one of 256 possible motor positions. Each time the motor is stepped, the motor position counter will be incremented or decremented, depending upon the direction of motor rotation. These software features allow the active control system to be completed.

Power-on reset will initialize the segmented registers with estimates of the motor position count for each oxygen segment. Furthermore, power-on reset drives the motor to the fully open position as determined by a mechanical limit switch. The motor position counter will then be initialized. After system sensors are read, the direction of rotation is determined as previously explained. A software test is then made to determine if actual oxygen concentration differs from the desired concentration by more than 5 percent.

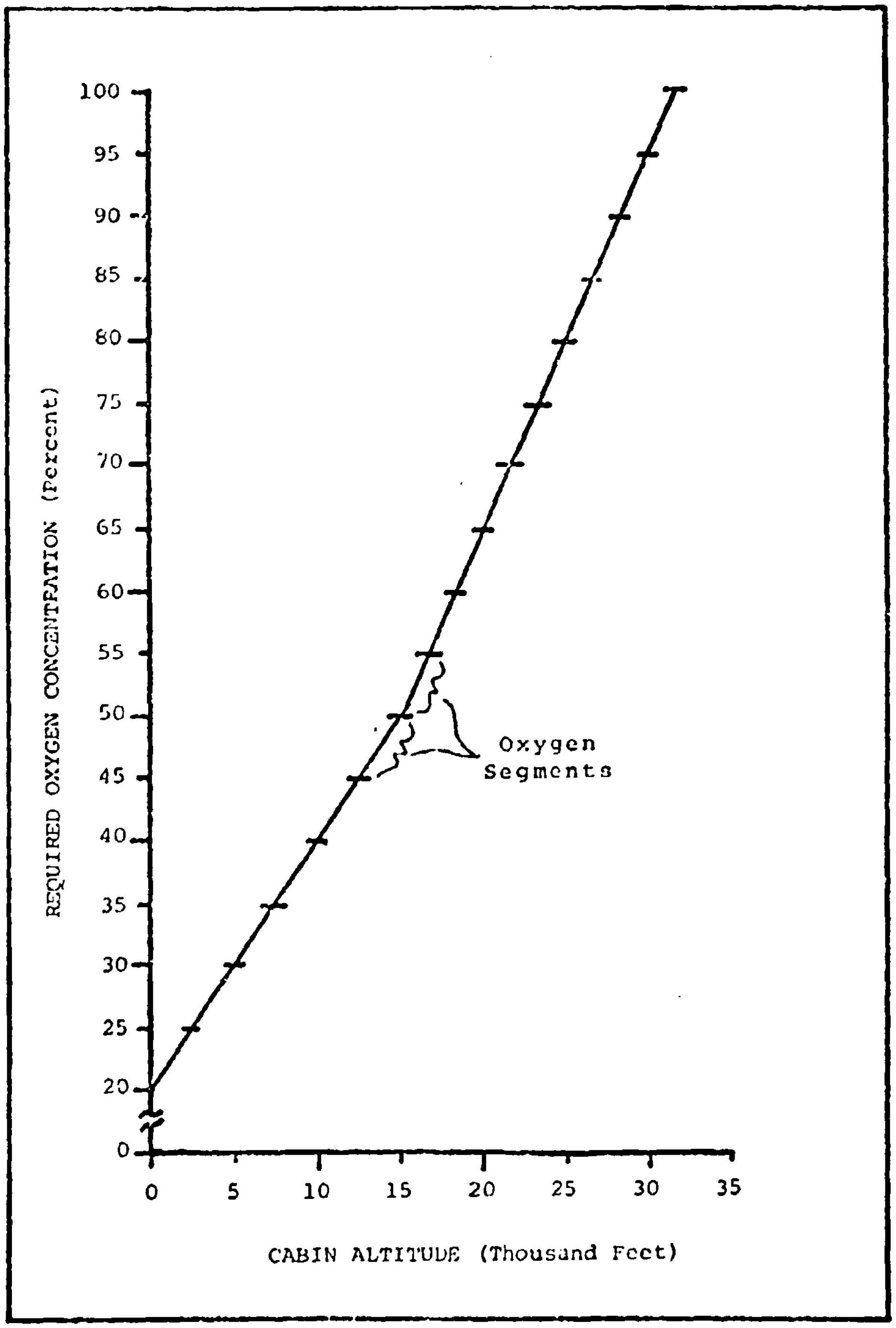


Figure 2. Segmented Physiological Oxygen Schedule (Ref 7).

If so, the motor is driven at the maximum rate to the segmented position/value of the corresponding desired oxygen demand curve segment. The software then jumps to the end of the initialization routine and the entire process is continuously repeated.

If the difference between actual and desired concentration is less than or equal to 5 percent, the motor is incremented by one step. Each time the motor is stepped, the motor position counter is appropriately adjusted. The software will then jump to the end of the initialization routine and the process is repeated. Another feature of the active control system is the update feature.

The update feature is used to keep segmented values accurate. If the desired concentration equals the actual concentration, the present motor position count is placed in the current segment of operation. With this concept, the oxygen concentration should never be more than 5 percent in error.

Summary of Results

The above hardware components and software concepts were integrated in the laboratory, and a prototype system was fabricated. Actual hardware components of a flight-testable prototype and a stepper motor were used. As requested by the SAM, potentiometers were used to simulate OBOGS sensor outputs.

Until an actual oxygen sensor is acquired, the system cannot be completely tested in an actual environment. Therefore, the system was tested in a laboratory environment. A set of digital displays was used to verify system operation. The displays indicate altitude and oxygen potentiometer settings in a digital format. The digital displays and potentiometer sensors allow system performance to be verified.

Each part of the system's software was individually tested by careful manipulation of the potentiometers. By comparing stepper motor response with the digital displays, system performance is ascertainable. The system performs as desired.

Overview

The remainder of this thesis contains a more detailed explanation of the OBOGS's control system. Chapter II discusses how an OBOGS generates oxygen and the requirements for a control system. The procedure used to design the control system is also discussed. Chapter III describes how hardware components and software procedures are integrated to form the control system. Chapter IV describes the results of the project and discusses the performance of the OBOGS's control system. Chapter V presents recommendations for a finalized, flyable control system and contains concluding remarks.

II. Design Procedure

Before any control system can be designed, the system to be controlled must be thoroughly understood. Therefore, this chapter contains an explanation of how the OBOGS generates oxygen. Next, a list of requirements as specified by the SAM is presented. With an understanding of the above topics, design of the control system can begin. The last part of this chapter discusses the procedure used to design the OBOGS's control system.

OBOGS's Operation

The functional diagram shown in Figure 3 will aid in a study of the OBOGS. Jet engine bleed air is used for the OBOGS air supply. A filter is used to eliminate dust, smoke, chemical warfare agents, and other contaminants. The filtered air is routed to a regulator providing a constant pressurized air supply for the molecular sieve bed. A regulator at this stage is used to reduce the pressure fluctuation in the bleed air supply, because the OBOGS requires a constant input pressure. The filtered air is routed from the regulator to a control valve (Ref 11:2).

The control valve is designed to alternate the air flow through the beds as follows. In one cycle, the air flows through the lower bed to the purge orifice and plenum, flows through the upper bed, and is then exhausted to the atmosphere. In the other cycle, air flows through the upper bed to the purge orifice and plenum, flows through the lower bed, and is then exhausted to the atmosphere. The physical properties of the molecular sieve bed require this cyclic air flow.

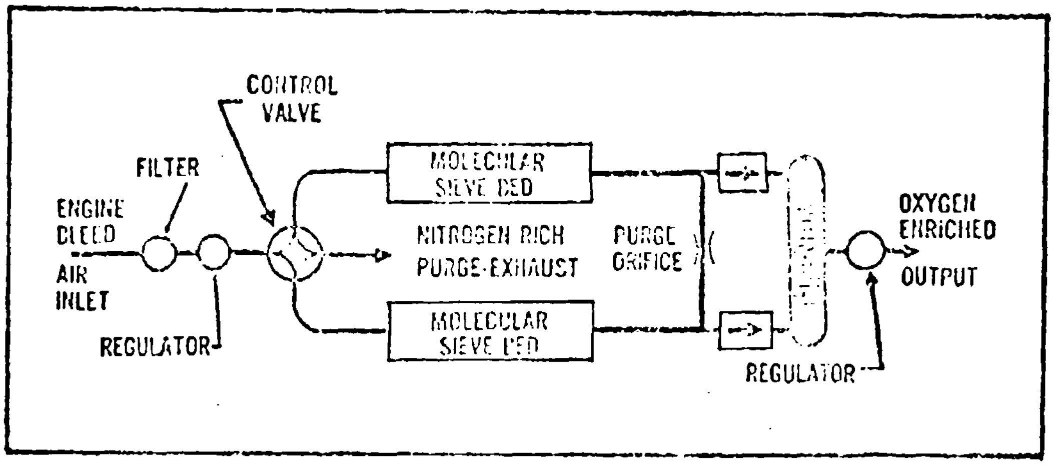


Figure 3. Functional Diagram of the OBOGS (Ref 11:3)

The properties of the molecular sieve bed allow oxygen to be extracted from air. A molecular bed is a chamber filled with aluminosilicate compounds called zeolites (Ref 11:4). Zeolites have a very uniform porous structure with pore sizes in the molecular range of oxygen, argon, and nitrogen. As air passes through the bed, zeolites preferentially absorb molecules because of weak Van der Waals forces. Large nitrogen molecules are absorbed and held longer than the smaller oxygen or argon molecules. Molecules of hydrogen, helium, and neon are so small that they migrate through the bed without significant absorption. As a wavefront of air passes through the bed, the wavefront separates into groupings of like molecules. Figure 4 shows the grouping of molecules after passing through this bed. The first molecules to appear at the output of the bed are hydrogen, helium, and neon. The molecules of argon and oxygen follow the first group of molecules. Nitrogen is the last type of molecule to appear at the bed's output (Ref 11:4).

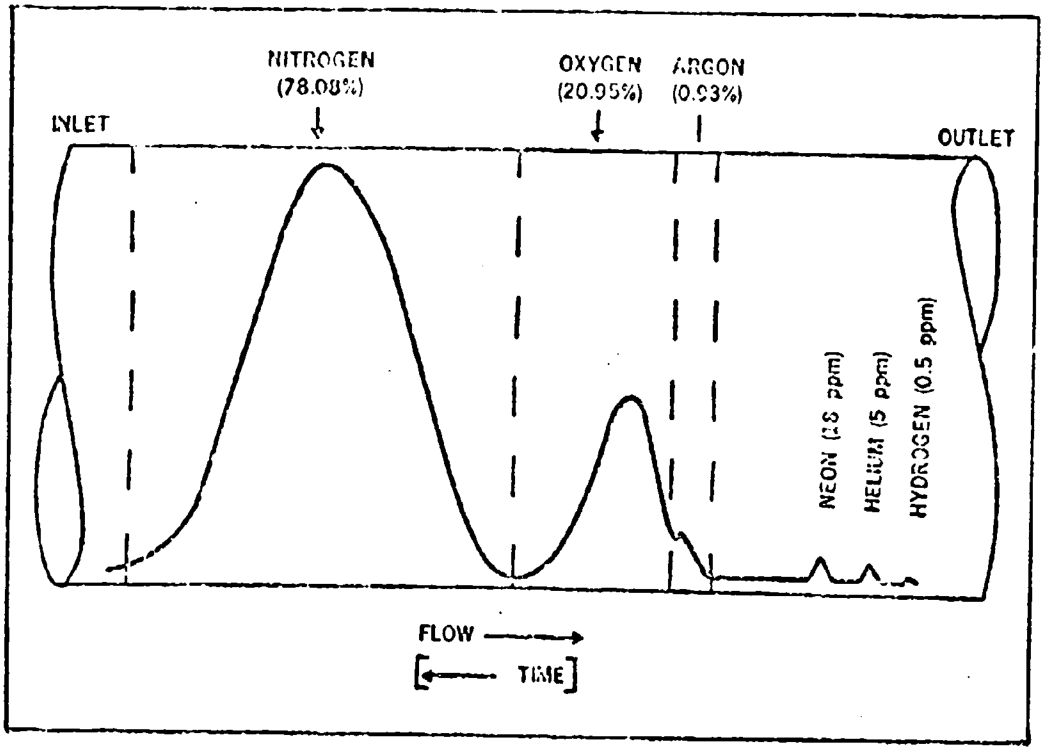


Figure 4. Idealized Air Separation of Molecules in the Molecular Bed (Ref 11:3)

Now the OBOGS is controlled to produce the desired concentration of oxygen.

Control of the OBOGS is accomplished with the control valve and purge orifice. If only one molecular sieve bed were to be used in a continuous mode, it would saturate with nitrogen and be incapable of further separation of gaseous molecules. Therefore, before saturation occurs the control valve is used to reverse the flow in the two beds. When a bed experiences a reverse flow, pressure in the bed is slightly decreased due to its new functional proximity to the purge-exhaust port. The pressure reduction during a back-flush is great enough to overcome the Van der Waals forces that attract nitrogen. This releases nitrogen from the bed. Nitrogen is then purged to the atmosphere, and the bed is able to reabsorb nitrogen when the cycle reverses. Further control of the OBOGS is accomplished by the purge orifice.

The purge orifice adjusts the rate of gas flow through the molecular bed. Proper adjustment of the purge orifice will allow a desired oxygen concentration to be generated. When the purge orifice is completely closed, the rate of flow through the bed is a minimum. At this minimum rate, the control valve reverses flow in the bed before nitrogen is released. Therefore, the maximum concentration of oxygen is obtained. When the purge orifice is completely open, more gas from the orifice manifold is allowed to escape to the atmosphere through the purge exhaust. Therefore, flow through the molecular bed is maximized. At the maximum rate, nitrogen is released from the bed before the control valve reverses flow. In this condition, a minimum oxygen concentration is generated. The purge orifice can vary the oxygen concentration in the purge manifold from 20 percent (normal air) to 95 percent (with the remaining 5 percent being argon) (Ref 11:4). The oxygen enriched gas in the purge manifold is extracted to produce an output from the OBOGS.

The check valves allow oxygen enriched gas to be extracted from the purge manifold. The check valves act as one way valves allowing gas to pass from the manifold to the plenum. A plenum holding tank is provided to keep back pressure on the check valves. Therefore, the check valves open only when pressure in the orifice manifold is greater than pressure in the plenum. Beyond the plenum, a regulator reduces pressure to a level acceptable for aircrew consumption.

Air flow cycle frequency in the OBOGS's beds is determined by the control valve's cyclic rate of operation. This rate has been optimized by the SAM and will not be altered. Therefore, adjustment of the OBOGS's output is controlled strictly by the purge orifice valve. The control system must adjust the purge orifice valve as specified by the SAM.

System Requirements

The School of Aerospace Medicine has presented several requirements for the OBOGS's control system. Those system requirements will be presented from a top-down, designer's viewpoint.

The overall system requirement is to design a control system capable of producing oxygen concentrations as specified by the physiological breathing schedule (Ref 7). The piecewise oxygen schedule, as presented in Chapter 1 (Figure 2), is to be used for the prototype model.

A more detailed requirement is to ensure that the physiological schedule be continuously maintained on a real-time basis (Ref 7). This requirement suggests that the control system response time should be held to a minimum. More specifically, the system should not contribute to reducing the time of useful consciousness. Since the time of useful consciousness varies from about three minutes at 10,000 feet to less than 15 seconds at higher altitudes, a one second or less response time appears to be an adequate goal. The controller should also maintain oxygen concentration within 5 percent of the desired oxygen concentration schedule (Ref 4).

Another requirement is to construct the system using digital components. It is also desirable to construct a system which allows future modifications to be added without complete redesign. Therefore, a programmable, microprocessor-based system is preferred (Refs 4, 7).

The SAM has also specified that system sensors are to consist of an altitude transducer and an oxygen concentration sensor. Rather than using actual sensors, potentiometers are used to simulate sensor response in the prototype system. Potentiometers allow the system to be tested in a laboratory environment in lieu of using altitude chambers. Also, the oxygen sensor has not been acquired; therefore, a substitute was mandatory.

Since an oxygen sensor has not been acquired, the system must be flexible enough to compensate for a range of possible sensor responses. Preliminary findings indicate that the oxygen sensor will have an extremely non-linear response. Also, the combined response time of the OBOGS and oxygen sensor may be in excess of 10 seconds (Ref 4). Therefore, the control system must compensate for these oxygen sensor shortcomings.

Design Implementation

The procedure used to design the OBOGS's control system was to match hardware components with software capabilities to meet system requirements. This section explains how hardware components were selected to implement the system. Chapter III discusses system hardware specifics as well as system software.

The design procedure discussed below is not intended to be a complete or detailed explanation describing every available alternative. The goal of this procedure was not to select the absolutely best choice of system components based upon exhaustive design criteria. Instead, the goal of this project was to demonstrate whether or not a digital control system is viable for the OBOGS. Therefore, more attention was given to producing a working prototype than to selecting optimum components. An effort was made to use reliable components of minimum cost and to minimize the number of components in the system.

Since adjustment of the OBOGS's oxygen conentration is accomplished with the purge orifice valve, a stepper motor was chosen for valve positioning. Stepper motors are ideal for valve control, and a variety of stepper motors are commercially available (Ref 3:73).

Stepper motors appear advantageous over other types of valve control hardware for several reasons. Incremental steps are small enough to achieve sufficient accuracy in positioning the valve. Also, strong holding torques keep the rotor stationary when rotation is not desired. While control of stepper motors is difficult with analog electronics, motor control is economically feasible with digital electronics (Ref 9:163). An optimum method of motor control is to use an IC stepper motor driver. This single chip generates the step sequence necessary for motor control and reduces the number of electronic components in the system. With the stepper motor driver, only two control signal inputs, rotation and trigger (R, T), are required for motor control.

Another system requirement was to use a microprocessor-based system. This requirement stems from the desired capability of being able to change operating characteristics without having to completely redesign the system. Therefore, a programmable system is required. Standard architectures for programmable digital systems include a central processor unit with external memory and complex support circuitry. A preliminary estimate of OBOGS's control system software indicated that less than 1 K words would be necessary for memory requirements. Since some MPU's contain internal memory, it is possible to construct a system using a single chip MPU with on-board memory. Therefore, a programmable system is best achieved with a MPU-based system. The MPU is capable of driving the motor according to any programmable procedure. Since a MPU operates at high speeds, control system response of one second or less can easily be obtained. MPU's are also available with 1 K word internal memories which permit the use of complex control algorithms that can compensate for non-linear sensor response and for the OBOGS's slow response time. The next task is to find a suitable MPU for this system.

When selecting a MPU to accomplish a task, several items must be considered. The MPU must contain an adequate instruction set and must execute instructions fast enough to accomplish the desired task. Also, the MPU must have an optimum word size architecture. MPU's are commercially available in 4, 8, and 16 bit word architectures. Larger size words provide higher bit resolution, but reduce the number of words available in a limited memory space. An 8 bit word provides a resolution of one part in 2^8 parts, or 0.39 percent accuracy. The 4 bit word gives 6.25 percent accuracy, and the 16 bit word gives 0.00153 percent accuracy. Since an overall system accuracy of less than 5 percent is desired, the 8 bit architecture is an optimum choice.

Of the many microprocessors available, Intel's 8035 MPU was selected for the prototype system. The 8035 is commercially available and is representative of other 8 bit MPU's. The 8035 operates from a single 5 V supply, contains over 90 instructions, and is readily available in military temperature versions (Ref 6:6-19). There are 27 available input/output (I/O) lines, including two software testable inputs in the 8035. This large number of I/O lines helps minimize the system's chip count.

A variety of memories are available in the 8035 family (called the MCS-48 family) which allow the designer to build a system using external random access memories (RAM), to breadboard a system using external or internal EPROM, and to use internal read-only memory (ROM) for production versions (Ref 5:1-2). For example, the 8048 MPU has a 1 K word x 8 bit internal ROM memory with a 64 word x 8 bit RAM scratchpad memory (Ref 5:1-3). The 8048 is pin compatible with the 8035, and the on-board memory is large enough for this system's software requirements. This capability reduces development cost, minimizes production costs, and reduces chip count. The 8035's internal data scratchpad memory eliminates external data memory and further reduces chip count.

An initial control system was built using an 8035 MPU with external RAM. Programs were stored in external RAM and executed using Intel's 8048 Control Computer (CC). The 8048 CC allows keyboard input to RAM with editing and debugging facilities (Ref 6). Also, the 8048 CC permits user access to the 8035 I/O lines. After the initial system was completed using RAM, the prototype system was completed using an 8035 MPU and EPROM. A single chip MPU and memory can be built (as discussed in Chapter V) which will further reduce system complexity.

The last major feature is to incorporate an input capability. Inputs to a MPU require a digital format even though system sensors produce analog voltage outputs. Therefore, an analog to digital (A/D) converter is a necessary interfacing component. Of the many available eight bit A/D converters, the National Semiconductor ADC 816 Single Chip Data Acquisition System was chosen. The 816 has 16 multiplexed input channels available with start and end of conversion control signals. The 816 is also significantly less expensive than other A/D converters.

Two other optional features were added to the control system. These features were not required by the SAM, but seem to be necessary from a system's viewpoint. To allow inflight monitoring of system performance, two sets of digital displays were provided. One display shows the desired oxygen schedule concentration (0 - 99 percent), and the other display shows actual oxygen concentration (0 - 99 percent).

The other optional feature was an emergency switch. When activated, the emergency switch drives the motor to the fully closed position, and the OBOGS generates maximum oxygen concentrations. This feature bypasses system sensors, the A/D converter, and the MPU.

Only the motor, IC driver and a few support chips are necessary for emergency switch operation. Thus, the emergency switch provides a back-up capability in the event of an inflight failure of the MPU, EPROM, A/D converter or system sensors.

This chapter has described the function of the OBOGS and presented the SAM's requirements for the digital control system and rationale for selection of the major components used in the control system. However, to completely understand control system operation, a more detailed explanation is necessary. The following chapter will describe how individual components operate, and it will discuss how the MPU is programmed to control system components.

III. Specifics of the Control System

This chapter describes operating characteristics of the major components used in the control system. Then, a layout of control lines and data paths is presented. Finally, the control system software is discussed.

Hardware

Major components discussed in this section include the stepper motor, its IC driver, a discrete transistor interface circuit, the MPU, the analog to digital converter, the digital displays, the limit switches, and the emergency switch. A presentation of these components is included for those readers who may be unfamiliar with these topics.

Stepper Motor and IC Driver. Unlike conventional motors, stepper motors have the ability to rotate in increments. These units, called steps, vary from motor to motor and have a typical range of 1° to 30° (Ref 10:36). Thus, a 1° step-angle motor could stop at any of 360 positions in a single revolution. Stepper motors can also run continuously in a clockwise (CW) or counterclockwise (CCW) direction. Stepping properties are possible due to the gearlike teeth around the periphery of the rotor (Figure 5). The magnetized rotor has an alternating north-south polarity on the gear teeth. The polarity on the rotor teeth creates torque on the rotor causing the rotor to align itself with the stator poles (Ref 2:94). The torque produced is a result of the magnetic principle in which like poles repel and

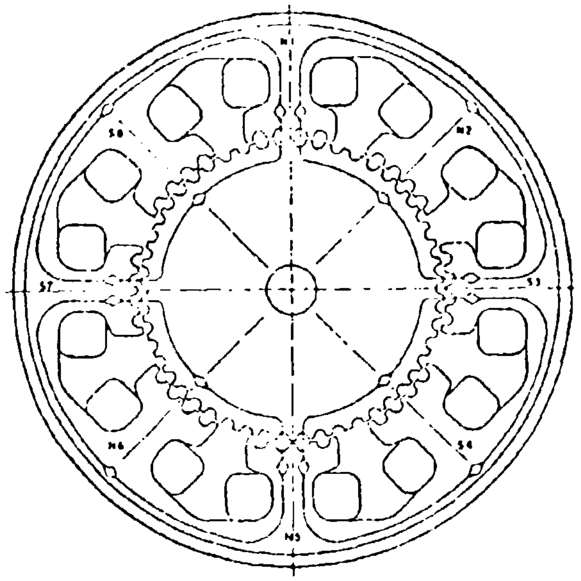
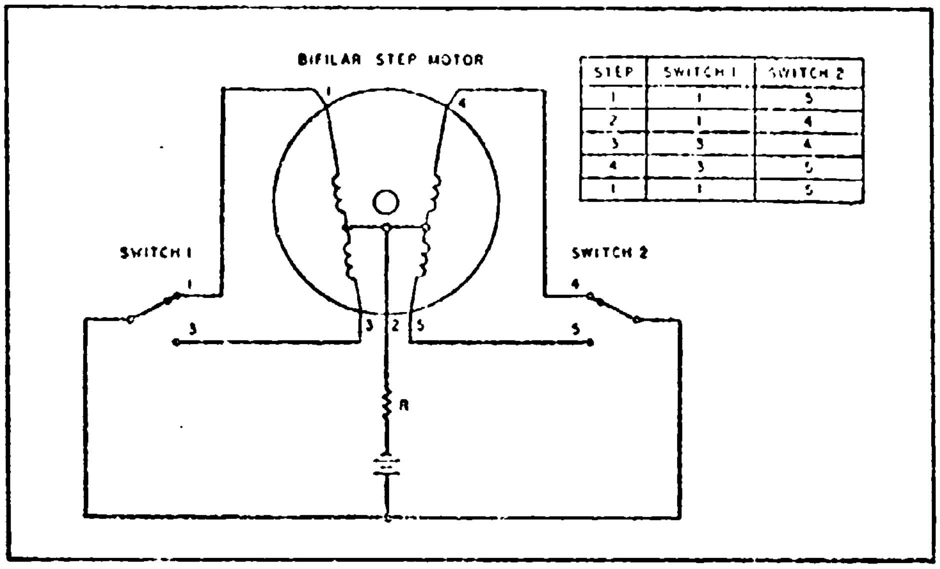


Figure 5. Stepper Motor Rotor and Stator Tooth Configuration (Ref 7:6)

unlike poles attract. In stepper motors, the stator poles are individually wire wound and individually controlled. The direction of current flow in a stator coil determines its magnetic polarity. Therefore, the direction of current flow in each stator pole will create CW or CCW rotor movement (Ref 14:5-7).

Figure 6 shows a functional diagram of a four pole, DC stepper motor. A four pole stepper motor requires two switches to control stepping action. In the number 1 step position, switch 1 is set to terminal 1, and switch 2 is set to terminal 5. If the switches are moved simultaneously to the second step position, the stepper motor will rotate one step CW. This process is repeated for each



STEP	SWITCH 1	SWITCH 2
1	1	5
2	1	4
3	3	4
4	3	5
1	1	5

Figure 6. Four Pole DC Stepper Motor and Step Sequence (Ref 2:94).

CW step. A clockwise step sequence is 5, 2, 3, 4, 1, 2, 3, 4, and so on. To reverse direction for a CCW rotation, a reverse step sequence of 4, 3, 2, 1, 4, 3, 2, 1, and so on is used (Ref 13:42). The motor's step rate is determined by the switching rate which can vary from a slow rate (one step per hour) to the motor's maximum run rate (typically several hundred RPM). Control of the stepper motor is functionally divided into two parts. A switching sequence must be generated for CW and CCW rotation, and the rate of switching must be controlled to adjust to motor speed (Ref 8:216).

Recent innovations allow the switching sequence to be generated by using digital techniques (Ref 9:163). In the present system, a North American Phillips Controls Corporation SAA1027 IC Driver is used to generate the switching sequence. This device (Figure 7)

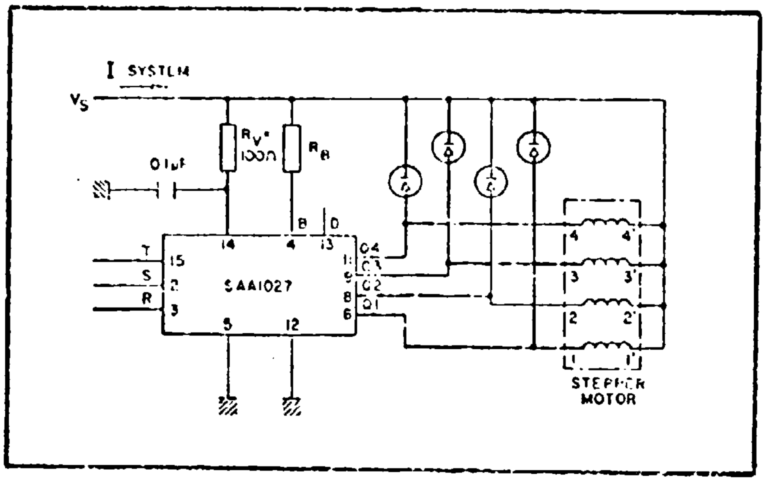


Figure 7. IC Stepper Motor Driver (Ref 1:CM822)

requires only two inputs (Ref 1). The rotation input (R) determines the CW or CCW direction of motor drive. The trigger input (T) determines the repetition rate of the switching sequence and ultimately governs motor step rate. The set input (S) is provided to initialize the switching sequence to motor step number one. This input is not used in the OBOGS's control system. Four outputs of the IC controller (driver) are connected to the stepper motor stator poles as shown in Figure 7. This single device reduces the amount of electronics needed to control the stepper motor. The stepper motor and IC driver operate on a 12 VDC supply voltage. Therefore, a 12 VDC supply is necessary for this system in addition to a 5 VDC supply for the other digital components. The IC driver R and T binary inputs require a 0 to 4.5 V input for the low logic level and a 7.5 to 12 V input for the high logic level. However, the MPU uses 0 V or 5 V for the low and high voltage logic levels. Therefore, an interface is used between the MPU and IC driver.

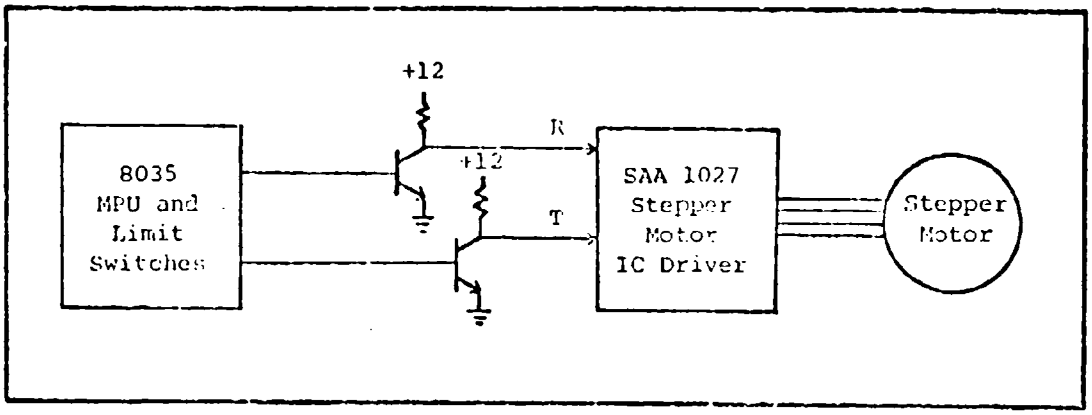


Figure 8. Transistor Interface

Transistor Interface. The transistor interface amplifies the output of the MPU's high voltage level of 5 V to approximately 12 V to meet IC driver requirements. Two circuits are used, one for the R signal and one for the T signal (Figure 8). This circuit is inverting, so the MPU must be programmed for appropriate logic levels. As the IC driver is controlled by the MPU, a discussion of the MPU is presented next.

Microprocessor. The MPU used in this system is the Intel 8035 shown functionally in Figure 9. Memory will be programmed with a set of event-related instructions. A complete discussion of system software is found later in this chapter. In general, instructions are available to read information from sensors, interpret that information, and to issue commands to peripheral equipment. Therefore, the sensor readings as decoded by the A/D, IC driver, and control lines must be connected to the MPU's input/output lines. Also, control lines are necessary for the A/D converter, memory, and digital readouts.

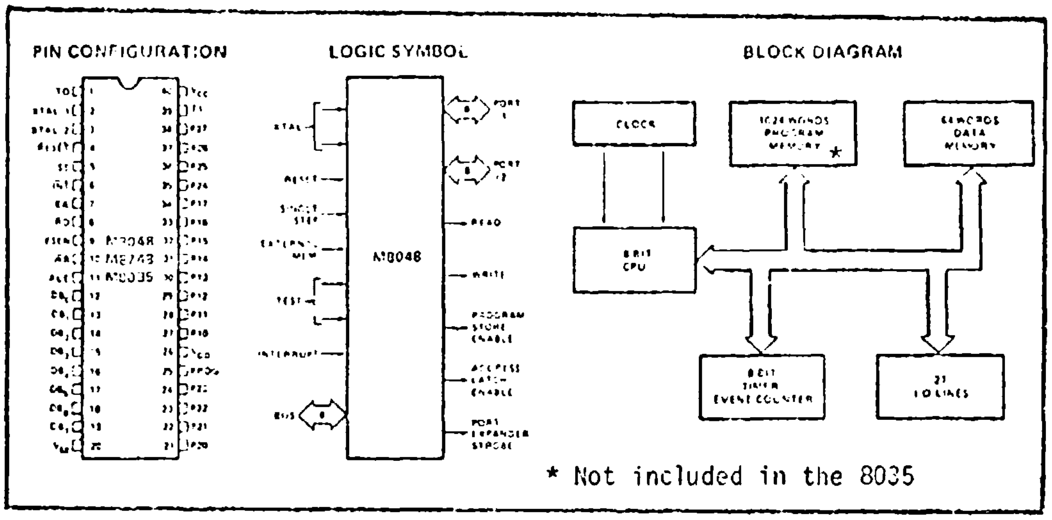


Figure 9. Intel's 8048 Microprocessor (Ref 5:6-19)

Resources of the 8035 include two eight bit, bi-directional ports which can be used for input or output and an eight bit, bi-directional bus which is used for the memory (Ref 5:2-4). The MPU also has a clock input, two testable inputs, and a reset input (Ref 5:2-5).

Allocation of the MPU's resources is as follows. The eight lines of port 1 are used to read sensor data from the A/D and to send data to the digital readouts. The eight lines of port 2 are designated as outputs and are used to control the stepper motor IC driver, A/D converter, and digital displays. The two test inputs are connected to limit switches which are driven by the stepper motor and are used to signal the MPU that the purge valve is fully open or closed. The reset switch is connected to the system's 5 VDC power supply. When power is turned on, the MPU will reset its program counter and begin executing instructions starting with address 000.

External EPROM memory is connected to the 8035 with an address latch and several control lines (Ref 6). Two 74174 latches are used to hold an address for the EPROM. In a memory fetch instruction, the program counter is output to the bus and to the lower half of port 2 (Ref 5:3-1). Address latch enable (ALE) from the MPU is used to signal the latch that the address is valid and the address is held in the latch. The EPROM is then given a program store enable (PSE) command from the MPU. The PSE command causes the EPROM to output memory contents to the MPU via the bus (Ref 5:3-1)

Since 12 address lines are required for memory fetches, an 8243 I/O port expander is necessary. This expander is software controllable and allows the lower half of port 2 to be expanded (Ref 5:3-6). With the 8243, the first four lines of port 2 are used as a data bus during memory fetch operations and are used as programmed I/O during other MPU operations. When port 2 is used to control the A/D or decimal readouts, the first four lines are routed through port 4 of the 8243. This arrangement is shown in Figure 10. A 3.0 MHz crystal clock provides necessary timing commands for the MPU.

Analog to Digital Converter. The ADC 816 converter is used to convert the analog voltage sensor output to a digital, 8-bit binary representation. Functionally, the A/D converter is shown in Figure 11.

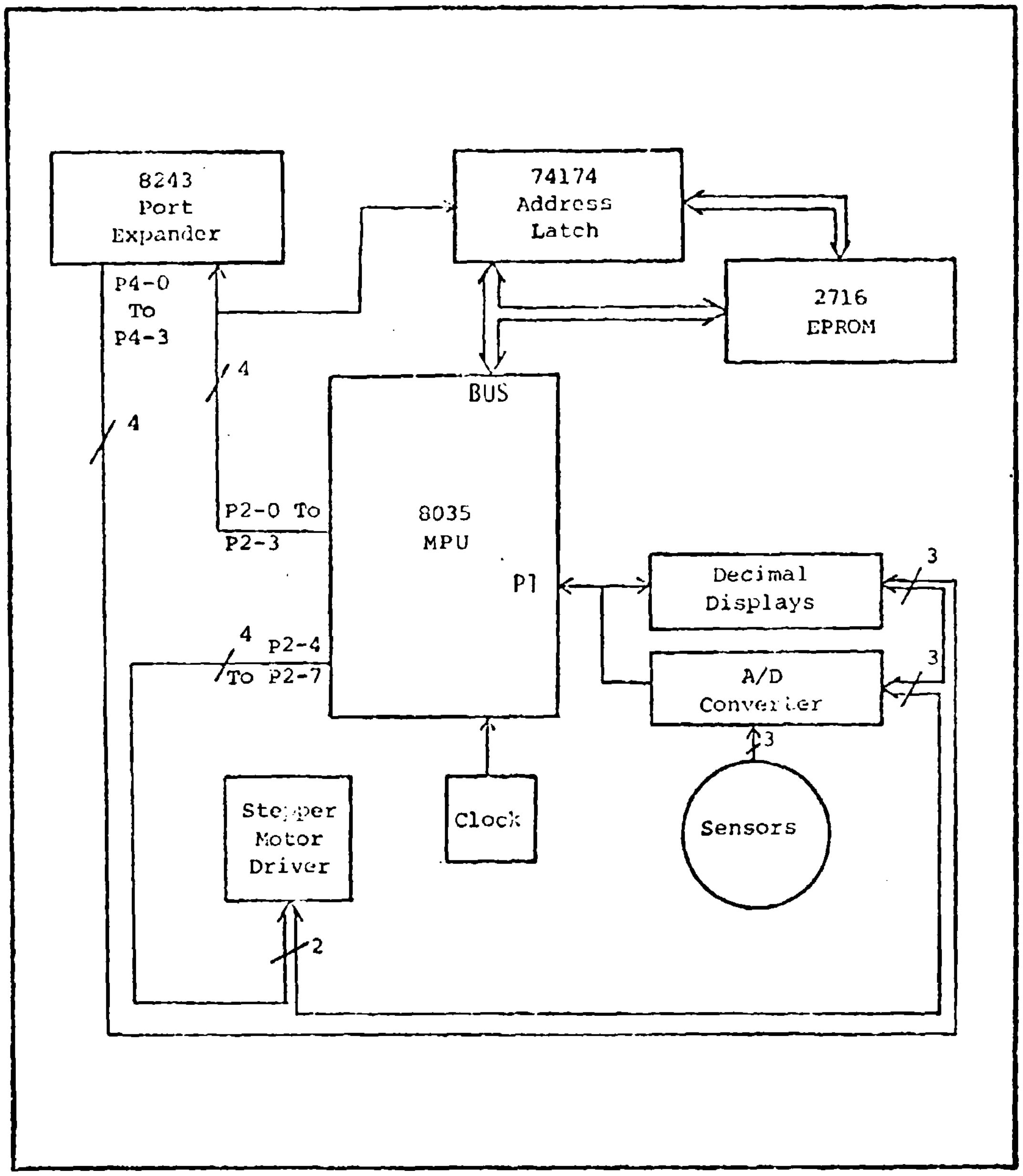


Figure 10. Control System with External Memory and Expander.

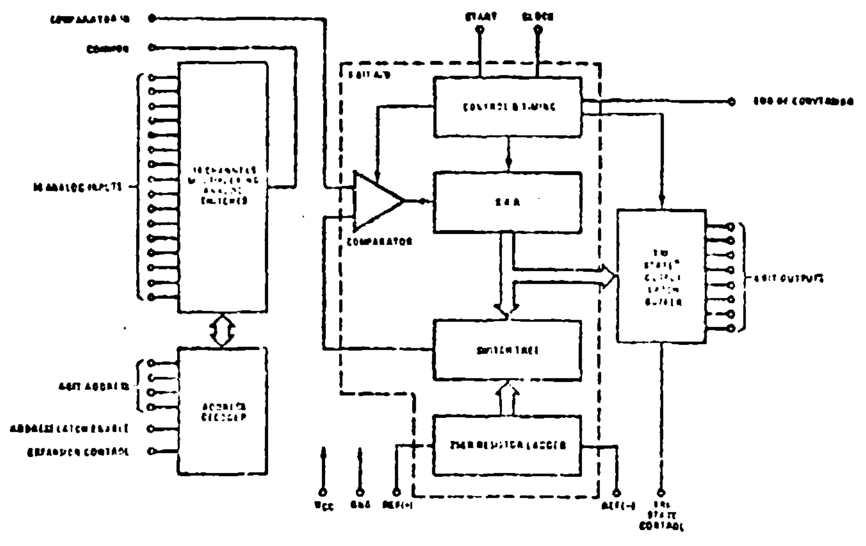


Figure 11. Analog to Digital Converter (Ref 12:1-184)

The 816 can convert up to 16 different inputs (Ref 12). This system uses three inputs; therefore, two address lines must be provided by the MPU. Lines P2-6 and P2-7 are allocated for this purpose. These lines will be software controlled to select a channel for conversion. After the address lines are set, the MPU's line P2-1 (P4-1 from the expander) is used to latch the address and to start A/D conversion.

The 816 uses a successive approximation method which produces an error of less than one half of the least significant bit value (Ref 12). Upon completion of the conversion, an end of conversion line is available from the 816. This line is not used in order to save MPU assets and to minimize chip count. Instead, a software delay is used and is adjusted for the longest possible A/D conversion time (eight clock periods) (Ref 12). When conversion is complete, the digital data is latched in the A/D output buffer. The actual software delay for A/D conversion is set for over 20 clock periods and is much greater than the A/D's 8 clock period requirement. This delay is used to assure A/D conversion is complete and to slow the main program's rate of execution. Therefore, the delay prevents the motor from overspeeding. This data is then read by the MPU under software control.

Digital Displays. The digital displays are also controlled by the MPU. Two sets of information are displayed, the desired oxygen concentration, and the actual oxygen concentration. This information is output from port 1 of the MPU to two sets of light emitting diodes (LED's). Each display contains 2 LED decimal digits so values from 00 to 99 can be displayed. Data is sent via software control to a binary coded decimal (BCD) converter which uses three 74185 chips.

Since the digital displays use the same MPU lines as the A/D converter, a 74126 bus gate has been added as shown in Figure 12. The MPU controls data flow in port 1 by using line 4-0. Setting P4-0 high allows data to flow from the A/D to the MPU. Setting P4-0 low blocks the path from the MPU to the A/D and enables the BCD converter. Data from the MPU is sent to both sets of LED's. Lines P4-2 and P4-3 are used to latch the appropriate LED display. When P4-2 is low, the desired oxygen concentration display is enabled. When P4-3 is low, the actual oxygen concentration display is enabled.

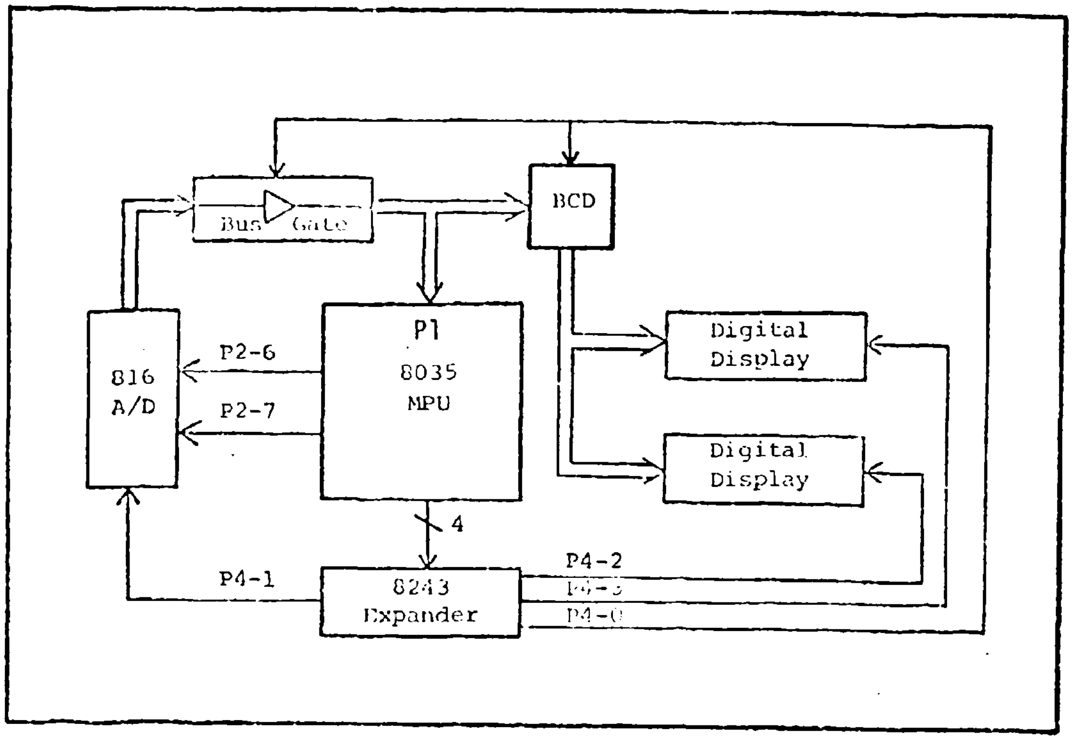


Figure 12. Control System with Bus Gate, BCD, and Digital Displays.

Limit Switches. As previously mentioned, limit switches have been provided to indicate when the purge orifice valve is fully open or closed. These switches are mechanically set according to the valve's limit of travel. The switches are connected to the MPU's two test inputs (T0 and T1) as shown in Figure 13.

Limit switches provide the following functions. At power-on reset, the motor is driven to the fully open position which is determined by the open limit switch. The MPU is programmed to test for a 5 V signal at input T0 for a fully open position indication. The closed limit switch works in a similiar manner. The upper half of the limit switches provide a mechanical back-up to keep the

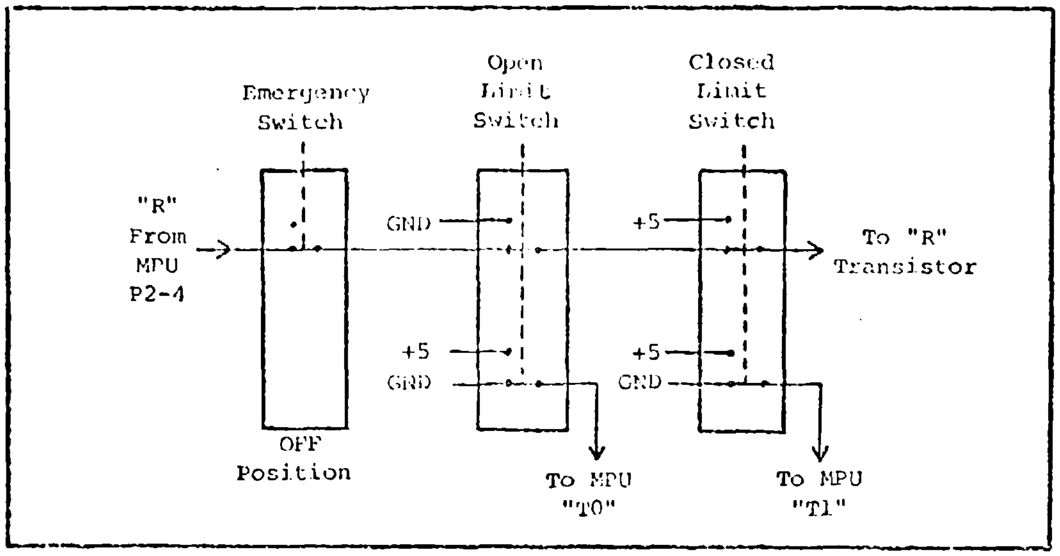


Figure 13. Limit Switches

motor from driving the valve past its mechanical limit. This prevents damage in the event of a failure of the MPU, A/D converter, or system sensors. These switches mechanically reverse the rotation polarity to prevent the motor from driving past preset stops.

The rotation switch is routed through the emergency switch. With the emergency switch in its normally off position, the rotation line passes through the emergency switch and is unaffected.

An earlier version of the limit switches, similar to the circuit of Figure 13, had the "T" line routed through the switches. This system allowed "T" to pass through the switches unless a limit was encountered. While this system was less complex, it required a manual repositioning of the limit switch, so future "T" inputs could increment the motor. This undesirable feature was eliminated with the circuit of Figure 13.

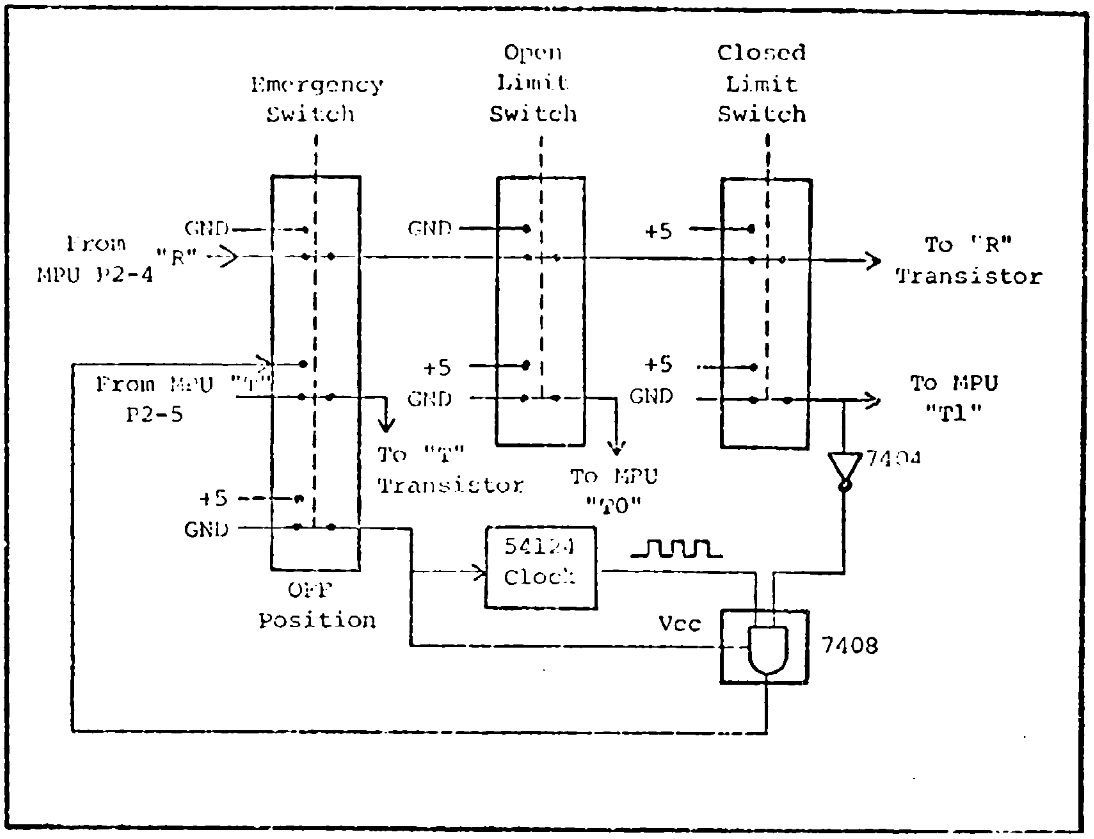


Figure 14. Emergency Switch System

Emergency Switch. The emergency switch allows the crew member to select maximum oxygen concentration when flight conditions warrant. Under normal operation, any changes in required oxygen concentrations will be controlled by the system. However, in the event of system malfunction, the emergency switch can be used as a back-up. The switch bypasses the MPU, memory, and A/D converter. Only the motor, IC driver, transistor interface, limit switches, and emergency switch need to be intact for successful emergency switch operation.

The emergency switch system is shown in Figure 14. When the emergency switch is turned on, the following sequence of events occurs. Power is applied to the AND gate and clock. The clock is a voltage controlled oscillator set by an external capacitor to oscillate near the motor's maximum run rate. Therefore, the clock drives the T transistor and in turn, the IC driver. Direction of rotation is also set by the emergency switch and drives the motor to the closed position. As the closed limit switch is set, the AND gate prevents clock pulses from triggering the T transistor. At this time, the stepper motor's holding torque can hold the valve in the closed position until the emergency switch is turned off. Normal system operation resumes if the emergency switch is turned off.

This completes a discussion of major system hardware components. With the hardware circuits and control paths established, the MPU could be programmed. Before the software is discussed, a review of the control lines is presented to assist the reader in understanding the software.

Control Lines and Data Paths

With the hardware basics presented, the next element in the design is the software. One important function of software is to issue control signals at proper times to various system components. Therefore, to assist the user in comprehending the software, the circuit of Figure 15 is presented.

Figure 15 indicates data paths and control paths of the entire system. Also, labels indicate which MPU I/O lines are used to control various system components.

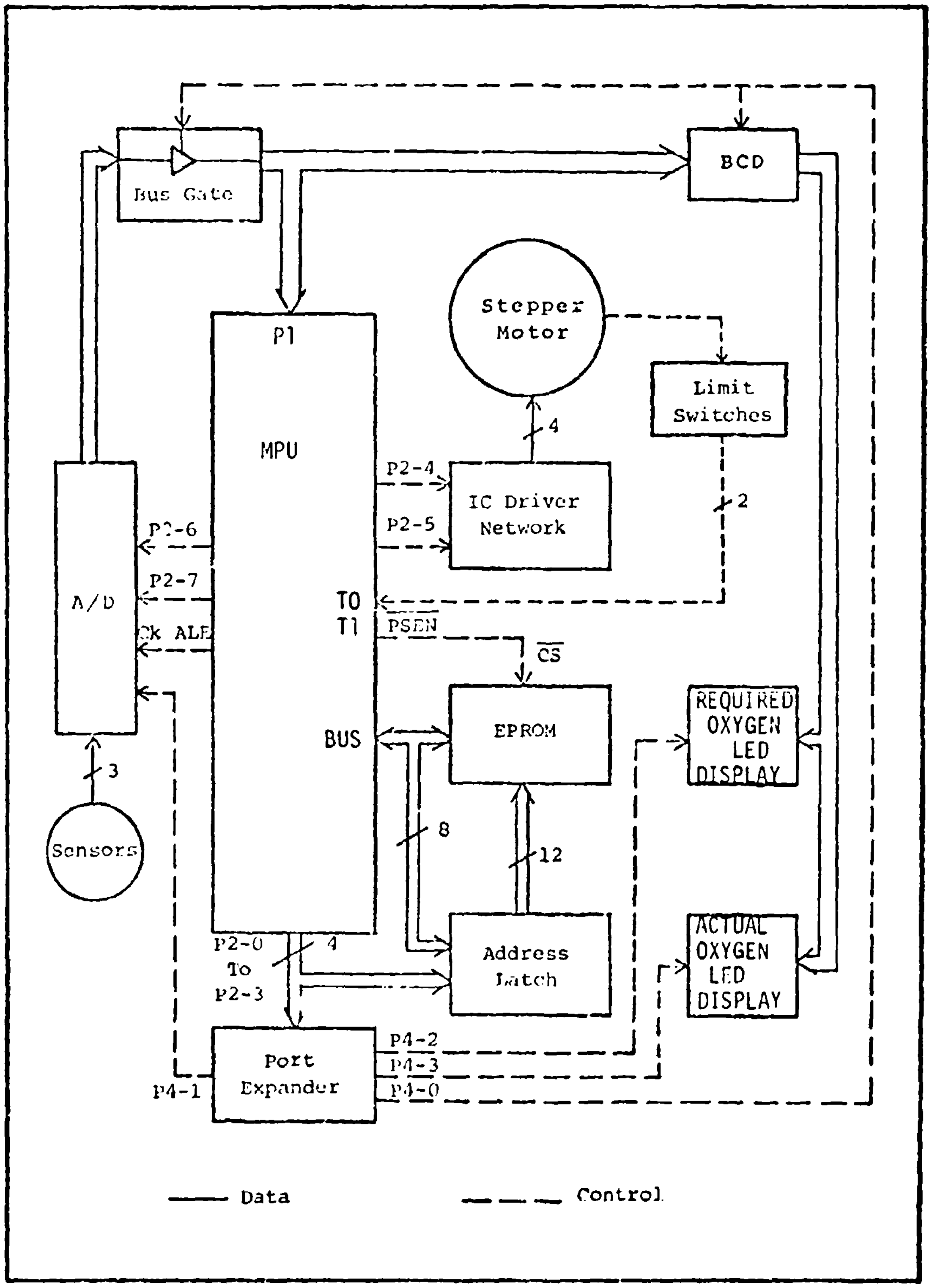


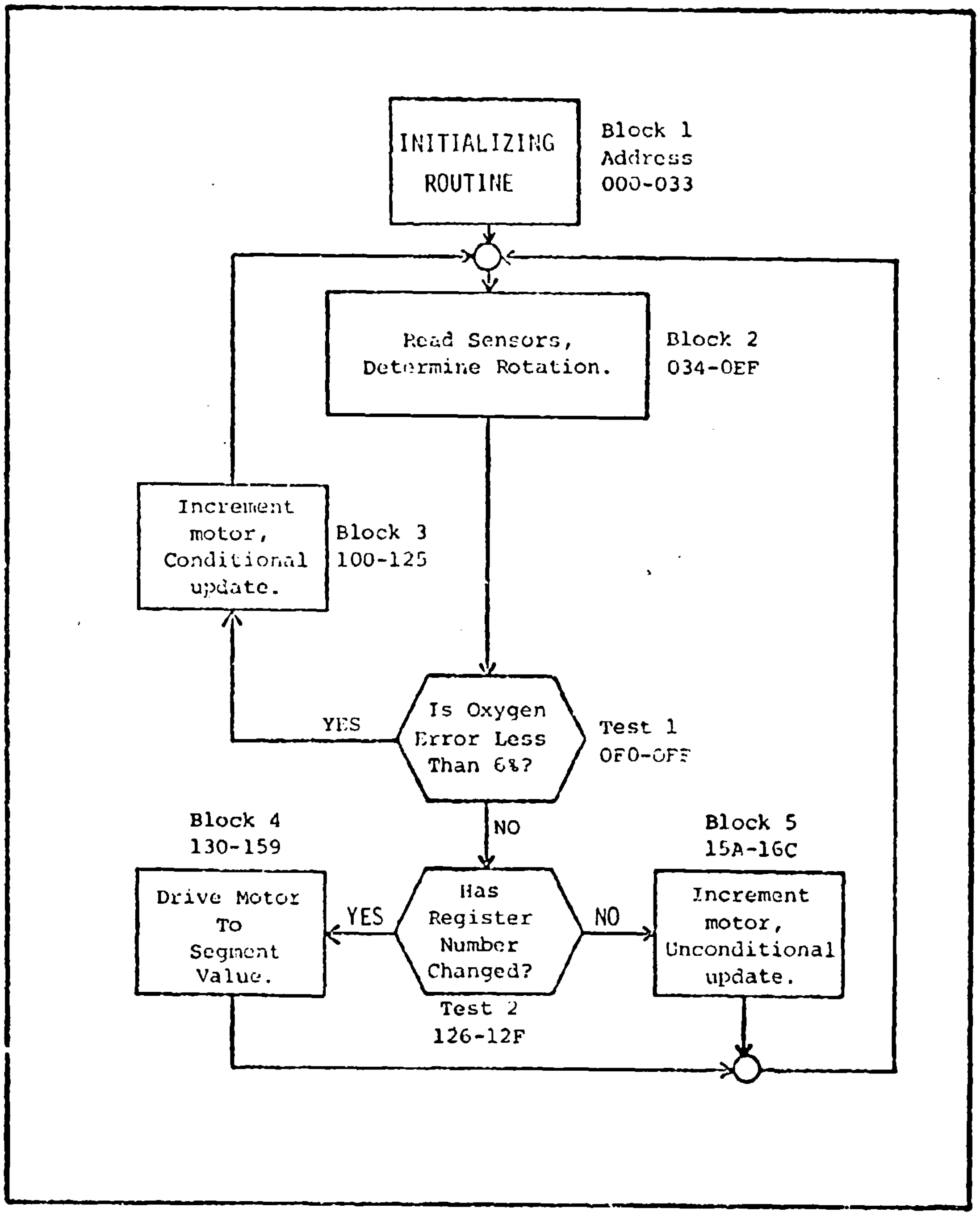
Figure 15. Control Lines and Data Paths.

Various circuits were constructed before the final prototype of Figure 15 was completed. Earlier versions were more complex and required more than the eight available control lines of port 2. This earlier system used 74125 and 74126 bus gates to multiplex control lines. A single line enables the bus gates which can direct up to four control lines to one of two different sets of destinations. This method was successful, but was not required in the final prototype control system. However, if future modifications (Chapter V) require additional control lines, a bus gate system could be incorporated.

Software

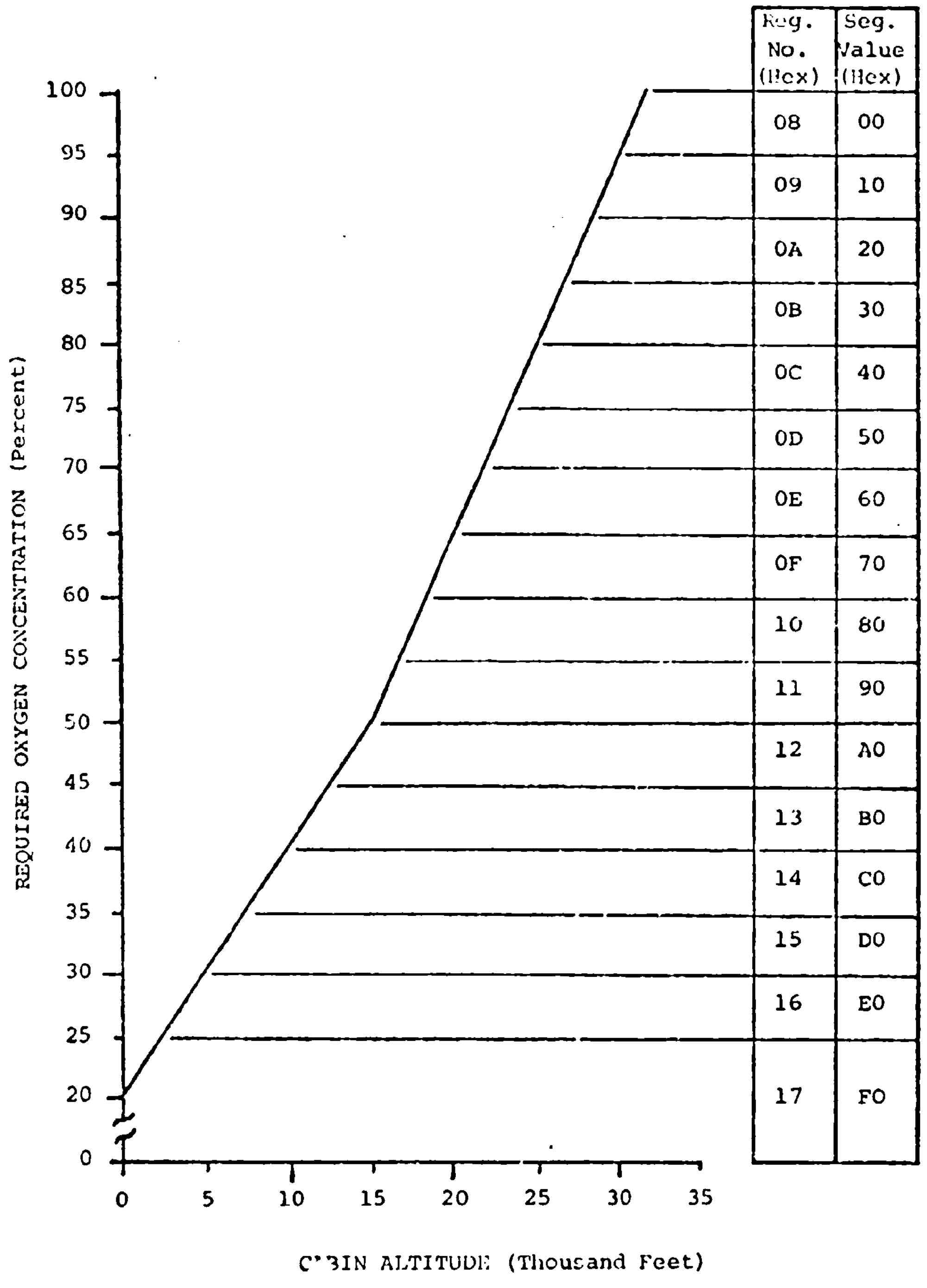
As previously discussed, the OBOGS's control system uses hardware or software as appropriate to accomplish individual system requirements. The software is a collection of instructions which allows the MPU to read data from sensors, interpret that data, and to issue control signals to drive the stepper motor. The instructions used for the 8035 are in assembly language form. Since assembly language is rather lengthy and tedious, the listing will be reserved for the Appendix. It is more meaningful to discuss the software flowchart in this section. However, as the flowchart is discussed, references to assembly language addresses are included so the reader may correlate the source code to the software flowchart.

The system flowchart is presented in Figure 16 and shows only main events. To assist the reader in following the software discussion, an altitude schedule is also presented. The altitude



INITIALIZING ROUTINE
Block 1 Address 000-033
Read Sensors, Determine Rotation.
Block 2 034-0EF
Increment motor, Conditional update.
Block 3 100-125
Is Oxygen Error Less Than 6%?
Test 1 0F0-0FF
YES
NO
Block 4 130-159
Drive Motor To Segment Value.
YES
Has Register Number Changed?
NO
Test 2 126-12F
Block 5 15A-16C
Increment motor, Unconditional update.

Figure 16. System Flowchart.



Reg. No. (Hex)	Seg. Value (Hex)
08	00
09	10
0A	20
0B	30
0C	40
0D	50
0E	60
0F	70
10	80
11	90
12	A0
13	B0
14	C0
15	D0
16	E0
17	F0

Figure 17. Altitude Schedule with Initialization Values.

schedule in Figure 17 contains a table which correlates register number and initial register value to the partitioned oxygen segments. Discussion of the flowchart will be organized by the major flowchart blocks. The reader may wish to review the concept of system control as explained in Chapter I before proceeding.

Block 1. The initializing routine occupies addresses 000_{16} to 033_{16} of the system program. This routine is automatically entered with power-on reset. The first function of this routine is to set the programmable I/O expander. Accordingly, port 4 is set up as an output port (steps 000_{16}-002_{16}). Next, the digital displays are latched to prevent unwanted data from being displayed. This is accomplished by placing a high level on lines P4-3 and P4-2.

The motor is then driven to the fully open position. A series of R=0, T=1 and R=0, T=0 will increment the motor to the fully open position. These commands are issued and a software delay is then executed to limit the motor increment rate. This delay is necessary because the MPU issues commands faster than the motor can respond. The motor is incremented until the open limit switch is activated (steps 006_{16} to 018_{16}). The MPU register which is used to keep track of 1 of 256 possible motor positions (register R7) is then set to 255 (FF_{16}). This value corresponds to the fully open motor position.

Segmented table values are then loaded with initial values (steps $01E_{16}$ to $02F_{16}$). The altitude schedule is divided into 5 percent segments. Each segment has a corresponding reserved register (registers 8_{16} to 17_{16}) in the MPU. This routine initializes these registers with the values shown in Figure 17. Upon termination of this sequence, register 08_{16} contains 00_{16}, register 09_{16} contains 10_{16}, register $0A_{16}$ contains 20_{16}, and so on.

A later part of the program tests to determine whether the altitude register number has changed. Therefore, the initial register number is set to 00 (steps 030_{16} to 033_{16}). This ensures that test number two is passed the first time it is encountered after initialization.

Block 2. This block (addresses 034_{16} to $0EF_{16}$) begins with a routine that reads the delay potentiometer. This is initiated by selecting the A/D delay channel. Next, the A/D converter is given a command to start conversion (steps 037_{16} to $03C_{16}$). Then, a software delay has been included (steps $03D_{16}$ to 041_{16}) which allows the A/D to complete the successive approximation conversion. In order to save MPU I/O lines, a software delay is used in lieu of using the A/D end of conversion signal. Finally, the digital representation of the delay potentiometer value is read into the MPU via port 1. This value is used to form a software delay.

The software delay (address 048_{16} to 054_{16}) uses a triple nested loop in which the value from the delay potentiometer is repeatedly decremented. The purpose of this loop is to delay the MPU by a variable time span as determined by the delay potentiometer. The delay potentiometer is used to compensate for a variable system response time. Delay times from microseconds to over three minutes are possible with this method.

If the delay potentiometer is adjusted to the minimum setting, the delay loop provides minimal delay. In this case, the program could issue motor increment commands at a rate which exceeds the motor's maximum run rate. Therefore, another delay is needed to keep from overspeeding the motor. This delay was added to the delay time for the A/D delay channel conversion. Consequently, the delay constant is longer in the delay channel conversion than in other A/D delay routines.

The next routine of Block 2 is one which tests the limit switches (055_{16} to $05F_{16}$). In case the motor is driven to the stops, this routine is added to reset the motor position counter. Therefore, each of the MPU's testable inputs (which are connected to the limit switches) are tested, and the motor position counter is reset to a high or a low count as appropriate.

At this point, the altitude potentiometer/transducer is read by the MPU. This routine (060_{16} to $06F_{16}$) is similar to the routine that was used to read the delay potentiometer except that the A/D altitude channel is selected in this case. The value of the altitude potentiometer, which can be 00_{16} to FF_{16}, is used as an entering argument for the altitude schedule table. The altitude table look-up program (070_{16} to 074_{16}) jumps to the altitude table (200_{16} to $2FF_{16}$) and returns to the main program with a value. The pre-programmed value is used to convert the altitude transducer output to the corresponding physiological oxygen concentration requirement. This table is used to account for non-linear sensor responses. The altitude schedule value is output for visual display to the LED's

(075_{16} to $07C_{16}$). This is accomplished by clearing the altitude schedule LED latch, outputting the schedule value, and then resetting the LED latch.

The oxygen sensor is now read by the MPU. This process is similar to the altitude sensor process except the appropriate control lines must be selected. This routine ($07D_{16}$ to $09F_{16}$) also displays actual oxygen concentration on the oxygen LED's and uses a separate table (300_{16} to $3FF_{16}$) for the oxygen sensor.

The next routine of Block 2 is to determine the oxygen schedule register number. This routine ($0A0_{16}$ to $0BF_{16}$) stores the last register number in an MPU storage location and determines the new register number of operation. These two values will be used in a later part of the program. The oxygen schedule register number is that MPU register number (08_{16} to 17_{16}) which corresponds to the segmented oxygen schedule. For example, if the altitude transducer is reporting 20,000 feet, the altitude schedule calls for approximately 44 percent oxygen. This falls in the 40 to 45 percent oxygen segment. The corresponding MPU register for this segment is register number 13_{16}. A series of subtractions and comparisons is used to determine register number in this routine.

The final task of Block 2 is to determine the direction of stepper motor rotation. This routine ($0C0_{16}$ to $0EF_{16}$) uses a subtraction and carry-test method to determine rotation. The actual oxygen concentration is subtracted from the desired concentration which influences the carry bit and which produces a magnitude called oxygen concentration error. The carry bit is used to determine clockwise or counterclockwise rotation which is output to the stepper motor IC driver. The magnitude or oxygen concentration error is used in the following test.

Test 1. The first software test is used to determine if a large error between actual and desired oxygen concentration exists. The purpose of this test is to reduce control system response time if large errors exist. For example, if a rapid decompression occurs, a large difference between actual and desired concentration will arise. The system cannot be allowed to act routinely by incrementing the stepper motor, delaying for the OBOGS's response time (possibly ten seconds), reading sensors, incrementing again, and so on, until the system is producing sufficient oxygen concentrations. This could easily exceed the aircrew's time of useful consciousness. Therefore, alternate procedures are used if large oxygen errors exist.

Test 1 ($0F0_{16}$ to $0FF_{16}$) determines if the difference between actual and desired concentration is less than 6 percent. For small errors, less than 6 percent, the motor will be incremented and the software will jump to the delay routine. The entire process is repeated until the actual and desired concentrations are equal. Incremental stepping of the motor occurs in Block 3. For large errors, additional computations are made in Test 2 which is discussed later.

Block 3. As previously mentioned, Block 3 (100_{16} to 125_{16}) is used to increment the motor if small errors exist between actual and desired oxygen concentration. The stepper motor is incremented by putting a high voltage level on line P2-4 which is connected to the trigger input of the motor driver. The trigger input is then removed for future increments. Each time the motor is incremented, the motor position counter is also updated. Software will increment or decrement the motor position counter depending upon the direction of motor rotation (107_{16} to 111_{16}). In this manner, the motor position counter is updated and can track one of 256 possible motor positions. With a 7.5 degree-per-increment stepper motor (48 steps per revolution), 256 positions can account for 5.33 turns. A 15 degree-per-step motor can therefore track 10.67 turns. The motor position counter will be used in a later part of this program.

The last function of Block 3 is called an update feature. This feature ($11A_{16}$ to 125_{16}) first determines if the oxygen concentration error is zero. If the error is not zero, a software jump causes control to be resumed at Block 2. If the actual oxygen concentration equals the desired concentration, an update occurs. The update places the motor position count in the segmented register corresponding to the current segment of operation. This feature updates the initialized segment value or last segment value with the most current value. A later part of the program will cause the motor to jump to the segment position. The update feature allows that jump to be as accurate as possible. The initial value is impossible to calculate due to the dynamics of flight and variable rates of oxygen consumption. Therefore, the update method was devised to keep the OBOGS operating within acceptable limits.

The remainder of this program is necessary to account for slow response times of the OBOGS and/or oxygen sensor. Several pieces of information have already been calculated for use at this time. The register number of the current and previous points of oxygen schedule operation have been determined, and the update feature has been incorporated. This information will be used starting with Test 2.

Test 2. The last portion of this program compensates for slow system response by rapidly driving the motor to a predicted position. This method is used if the incremental step method of Block 3 is expected to be too time consuming. A figure of five system response time periods has been chosen as the cutoff for incremental stepper motor corrections. If six or more time periods are required, Test 1 diverts program execution to Test 2 and Block 3 is bypassed.

At this time, it is desired to drive the m or at a rapid rate to a segmented position. However, another consideration must be included at this point. If the motor has just driven to a new segment position, and the error between desired and actual concentration is still large, an alternate path must be included. This situation could occur if the initialized segment values are in error by more than 5 percent or if the oxygen sensor response time is aperiodic. The oxygen sensor may have an oscillatory response if rapid fluctuations in oxygen concentration occur. Therefore, two control paths emanate from Test 2.

First, Test 2 determines if the segment/register number has changed between the current point of operation on the altitude schedule and the previous point of operation (126_{16}-$12F_{16}$). A change in register numbers indicates that the current oxygen segment is not the same as the segment determined by the last loop through the program. In this event, driving the motor to a new segment is appropriate as the oxygen schedule calls for operation in a new segment. Therefore, control is resumed with Block 4. Otherwise, a software jump causes execution to skip to Block 5.

Block 4. Software of Block 4 is written to drive the motor at a rapid rate to a new segment position. This block compensates for long OBOGS's response times as discussed under Test 2. The motor is incremented in this section (130_{16} to 159_{16}) until it has arrived at the new segment. Direction of rotation and the segment of operation was previously determined by Block 2. The motor is stepped to its new segmented position which is determined by comparing the motor position count with the segment value. The motor is incremented until the motor position count equals the segment value. Each time the motor is incremented, the motor position count is adjusted ($13A_{16}$ to 142_{16}) in a manner similar to Block 3. Since a software loop can increment the motor too rapidly, another delay loop is included (148_{16} to 150_{16}). This loop has a delay constant which has been adjusted for maximum motor speed. When the motor reaches the new segmented position, a software jump causes execution to resume at Block 2, and the entire process is repeated.

Block 5. This final block is used as a default route and is provided to account for inaccurate segment initialization or for possible erratic oxygen sensor readings as discussed under Test 2.

Operation in this routine will not occur after preflight of the OBOGS if a reliable oxygen sensor is demonstrated. Nevertheless, this block should be included to keep the system from entering a repetitive, nonproductive loop. Without this block, it would be possible to loop through Block 2, Test 1, Test 2, and Block 4 on a repetitive basis. In that event, a stalemate would occur until the oxygen sensor or cabin altitude changed by more than 6 percent.

When Block 5 is entered, the following conditions have occurred. An oxygen error of greater than 6 percent has been detected, and the motor could have been driven to a new segment. However, the motor cannot be driven to another new segment because the oxygen schedule still calls for operation in the same segment. The remaining alternative is to increment the motor in the direction as determined by Block 2. If a repetitive loop consisting of Block 2, Test 1, Test 2, and Block 5 occurs, eventually the register/segment number will change or the oxygen sensor will recover and the oxygen error will become less than 6 percent. Normal operation will resume at this time.

One other item is included in this block. It is possible that Block 5 was entered due to a large error in the segment value. To prevent this from recurring, the segment value is unconditionally updated whenever Block 5 is entered. A software loop jumps back to Block 2 at this point.

This concludes the functional description of the control system's hardware and software. Detailed circuit diagrams are contained in the Appendix, as well as a complete program listing. Chapter IV will present a discussion of the prototype and will discuss its performance.

IV. Results and Performance

This chapter describes the progress that has been made in constructing the OBOGS's control system. The next chapter will explain features that were not included in this initial design. Additionally, this chapter discusses system performance.

Results

A stand-alone control system was completed for this thesis project. The initial version of the system was built using an 8048 Control Computer with RAM. After this system was debugged, software was stored in EPROM. The EPROM version is a stand-alone controller requiring no peripheral support equipment with the exception of a power supply. This version also contains the decimal LED displays and the emergency switch. To adequately evaluate the system, a support bracket was fabricated. This bracket contains mounts for the stepper motor and mechanically adjustable limit switches. The support bracket can be mounted directly to the OBOGS's purge orifice valve. The system was constructed on a prototype board using universal sockets as requested by the SAM (Ref 4). When an oxygen sensor is acquired and tested, a flyable version can be constructed. After the prototype system was constructed, an evaluation of system performance was made.

Performance

Both hardware and software must be evaluated for dependability and functional performance. The hardware dependability for this system is as reliable as possible with commercial grade components. Hardware components consist of IC chips, switches and a stepper motor. IC chip technology has extended chip lifetime to a point where environmental factors are more likely to cause chip failure than intrinsic failure modes within the IC. The switches in this system are not toggled very often in an operational environment. Therefore, switch lifetime is not an area of major concern. Stepper motor lifetime is longer than that of conventional motors because heat buildup is 50 percent less in stepper motors (Ref 1). Consequently, hardware failure is expected to be minimized because the most reliable types of commercially available components were used in this system. However, a final design must be hardened against temperature extremes, supply voltage variations, and other aspects of aircraft environment.

The next topic of concern for system evaluation is performance. In this case, performance will be a measure of whether or not the system functions according to the system's specifications. Individual hardware components were separately evaluated and found to function according to the manufacturer's specifications. As software routines were developed, they were individually tested and functioned satisfactorily. After the system was completely constructed, the overall system was analyzed for performance.

Overall performance is best evaluated by testing each major loop of the software flowchart. As the system is tested, the two potentiometers were positioned for various input conditions while the decimal displays and motor response are observed. At turn on, the motor drives until the limit switch is activated. The digital displays then reflect values according to the potentiometer settings. At this point, motor response is related to the potentiometer settings. Each software loop is testable using the following procedures.

If the potentiometers are set for conditions that give less than 6 percent error between actual and desired concentrations, the first loop is entered. In this loop, the motor is incremented CW or CCW until the actual and desired oxygen concentrations are equal.

Any time the oxygen error exceeds 6 percent, the second or third loop is entered. If the oxygen error is due to a change in altitude, the motor is driven by the software of the second loop to the segmented position. This position is determined by either the initialized position or the updated position. Therefore, the update feature is testable.

The third loop is tested by setting the potentiometers for a 6 percent or more oxygen error and allowing the motor to drive to the new segmented position. If the oxygen potentiometer is not changed, the motor will increment until the oxygen error is reduced to zero.

The above tests were conducted several times to evaluate system performance. Each test was successful. Furthermore, there are no dead-end software paths that could lead to unfavorable responses. As the potentiometers were adjusted for various input conditions, there were no valid conditions which produced oscillations in the control system. Emergency switch operation can be activated at any time without affecting the software. Normal system operation resumes when the emergency switch is deactivated. The lack of an oxygen sensor precludes testing in an altitude chamber. However, this test is required more for the oxygen sensor than for the control system, since the software can compensate for nonlinear sensor response.

Successful results of the above tests indicate that the control system functions as desired. Every system requirement has been accomplished. The prototype system is complete with the exception of a power supply and sensors. Software is complete with the exception of the look-up tables which can be completed when sensor characteristics are provided.

Even though the system is complete, there are some optional features that can be incorporated at a later date. These ideas are discussed in the next chapter.

V. Recommendations and Conclusions

This chapter will present ideas for improving the capabilities of the final control system. A number of options became apparent during the system design phase which could not easily be included in the prototype system. Because some of these ideas may enhance the system's capabilities, they are presented in this document. This chapter also contains concluding remarks.

Recommendations

Although the present system design is complete, several options are envisioned for the final system. The first option involves the system sensors as shown in Figure 18. A cockpit selectable altitude potentiometer could be included for aircrew and maintenance personnel. The altitude potentiometer can be adjusted by maintenance personnel to assist in trouble-shooting possible malfunctions. When used by aircrews, the potentiometer allows a power on preflight to be accomplished which verifies the OBOGS's operation and updates initialized segmented register values with more accurate data. The potentiometer also gives the aircrew an inflight, manual back-up for the altitude transducer.

Another feature, as shown in Figure 18, is a back-up oxygen sensor. The back-up sensor is easily added to the system and could be included if sensor lifetime is unpredictable. Consideration can also be given to positioning the secondary sensor at a different physical location from the primary sensor. This may assist in

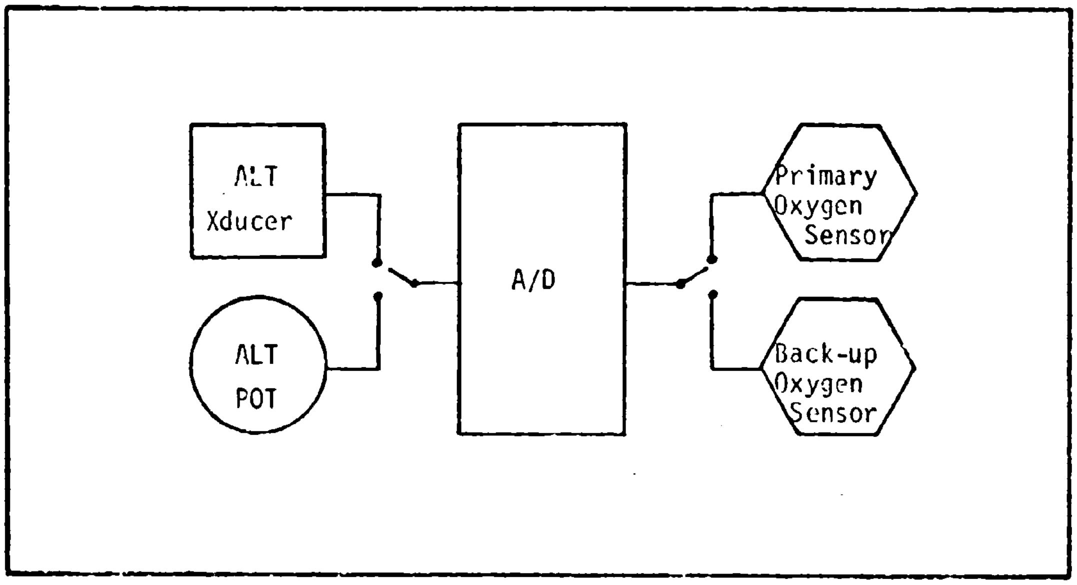


Figure 18. Optional System with Dual Sensors

configuring the OBOGS for both single- and multiple-aircrew aircraft.

To provide system built-in test equipment capabilities, a push-button switch could be added to the control panel. This could be connected to the MPU's interrupt input. When activated, the software could generate a series of functions which could test the digital displays, stepper motor, and other components. This feature could help determine whether or not the OBOGS is operationally capable without requiring aircraft engine operation.

Another optional feature is an alert system. An audio and/or visual alert could be incorporated which could be activated whenever rapid changes occur in required or actual oxygen concentrations. The specifics of this test depend upon the oxygen sensor's response. However, a possible method for determining an alert condition might be to monitor entry into Test 2 of the software flowchart. Aircrew response to an alert signal could include selecting the back-up oxygen sensor, activating the emergency switch, cycling the power switch, checking cabin pressurization, and checking for aircraft structural integrity.

After the system is finalized, several items must be considered during the packaging phase. The present system, with 16 IC chips, 2 transistors, and 4 LED displays, is mounted on a 6" by 8" circuit board. The 5 V supply must deliver 1.2 amps and the 12 V supply must deliver 0.75 amps. Regulation of the 12 V supply is not critical since the stepper motor and IC driver can operate from 9.5 V to 18 V (Ref 1). The 5 V power supply must be held within 4.5 V to 5.5 V, due to the operating limits of the MPU (Ref 5:6-19), A/D converter (Ref 12), and IC support chips (Ref 15). Therefore, if aircraft power supplies are not regulated within these limits at all times, a voltage regulator must be added. Temperature limits for the stepper motor and driver are -40° C to 85° C for storage and -20° C to 70° C for operation (Ref 1). The motor and driver should be located in an area where these limits are not exceeded. The most severe temperature limit for the remaining component is from the 2716 EPROM which has a permissible operating range of -10° C to 80° C (Ref 5:7-17). These limits should not be a problem since the entire circuit board area is small enough to be mounted in the cockpit. If an 8048 MPU with on-board memory is used, the required circuit area can be reduced to about 6" by 6", and the temperature range becomes -55° C to 125° C since the 2716, 8243, and 74174's are not required.

A one chip MPU and memory has additional advantages when considering reliability, production costs, and maintenance costs. Intel's 8048 MPU has a 1 K word by 8 bit on-board ROM which is pin compatible with the 8035. The present system software is 879 words long which can fit in the 8048's memory space. Also, an 8748 MPU with 1 K word by 8 bit EPROM is available. This allows the user to debug systems with EPROM before committing software to ROM versions. An on-board memory MPU system reduces the chip count by four and requires only minor software changes. Those changes include deletion of 8243 programming and changing output commands from port 4 to port 2.

One other possibility exists for further simplifying the system. Intel makes an 8022 MPU with on-board memory and a two channel A/D converter (Ref 5:6-36). The 8022 has a 64 word by 8 bit RAM and 2 K words by 8 bit of ROM. If the oxygen sensor and OBOGS's response time prove to be non-varying, this single chip system may be feasible. If the OBOGS's response time varies depending upon the number of crew members, perhaps an externally selectable input (set for response time) could be used and adjusted during preflight. In the event that more than two A/D channels are required, an analog switch could be used to multiplex A/D inputs, and a single chip system may still be realized.

Other features may be possible with this proposed OBOGS control system. However, the next major task is to acquire and test an actual oxygen sensor. Successful completion of this phase will justify efforts required to incorporate the above optional features into the system. Eventually, the system can be constructed on a printed circuit board and inflight testing can proceed.

Conclusions

When this project was undertaken, initial efforts were made to understand the OBOGS and the requirements for a control system. The concept of an active system control, using a segmented schedule, was then formulated. Hardware components were individually acquired and tested. System software was then written to control the system. This design procedure proved successful after a working prototype was completed. The on-board oxygen generator digital control system meets all requirements as specified by the School of Aerospace Medicine. As soon as an oxygen sensor is acquired, the system can be completed and additional features can be added which will further enhance the control system's capabilities.

Bibliography

1. Airpax. Stepper Motor IC Driver. Specifications, Form CM822. Cheshire, Connecticut, North American Phillips Controls Corporation, 1977.

2. Giacomo, Paul. "A Stepping Motor Primer," Byte Publication, Inc., 6: 90-105 (February 1979).

3. Harding, Bill. "Valve Control with a Microcomputer," Instruments and Control Systems, 102: 73-77 (October 1978).

4. Ikels, Dr. Kenneth G., Kolesar, Capt Edward S., Miller, Dr. Richard L., School of Aerospace Medicine, Brooks AFB TX. (Personal communications, 1980).

5. Intel Corporation. "MCS-48 Family of Single Chip Microcomputers," USERS Manual, Intel Corporation, Santa Clara, California, April 1979.

6. IMSAI Manufacturing Corporation. "8048 Control Computer," User's Manual, IMSAI, San Leandro, California, July 1977.

7. Kolesar, Capt Edward S., AFIT Thesis Topic Proposal, School of Aerospace Medicine, Brooks AFB TX, 1980.

8. Lafreniere, B.C. "Software for Stepping Motors," Machine Design, 51: 213-217 (April 26, 1979).

9. Lesea, Austin. "Applying Micros to Motor Control," Instruments and Control Systems, 102: 163-166 (September 1978).

10. Madsen, Elmer. "A Primer on Stepper Motors," Electro-Optical Systems Design, 31: 36-45 (June 1979).

11. Miller, Dr. Richard L., et al. "Molecular Sieve Generator of Aviator's Oxygen: Breathing Gas Composition as a Function of Flow, Inlet Pressure, and Cabin Altitude." Report SAM-TR-77-40, School of Aerospace Medicine, Brooks AFB TX, December 1977.

12. National Semiconductor, "ADC 0816 Single Chip Data Acquisition System," Data Sheet, IMB20M97, September 1977.

13. Seim, Thomas A. "Numerical Interpolation for Microprocessor-Based Systems," Computer Design, 15: 111-116 (February 1978).

14. Superior Electric. Design Engineer's Guide to DC Stepping Motors. Pamphlet. Bristol, Connecticut, Superior Electric Company, 1976.

15. Texas Instruments. The TTL Data Book for Design Engineers, Second Edition. Dallas, Texas, Texas Instruments, Incorporated, 1976.

APPENDIX A

CIRCUIT SCHEMATICS

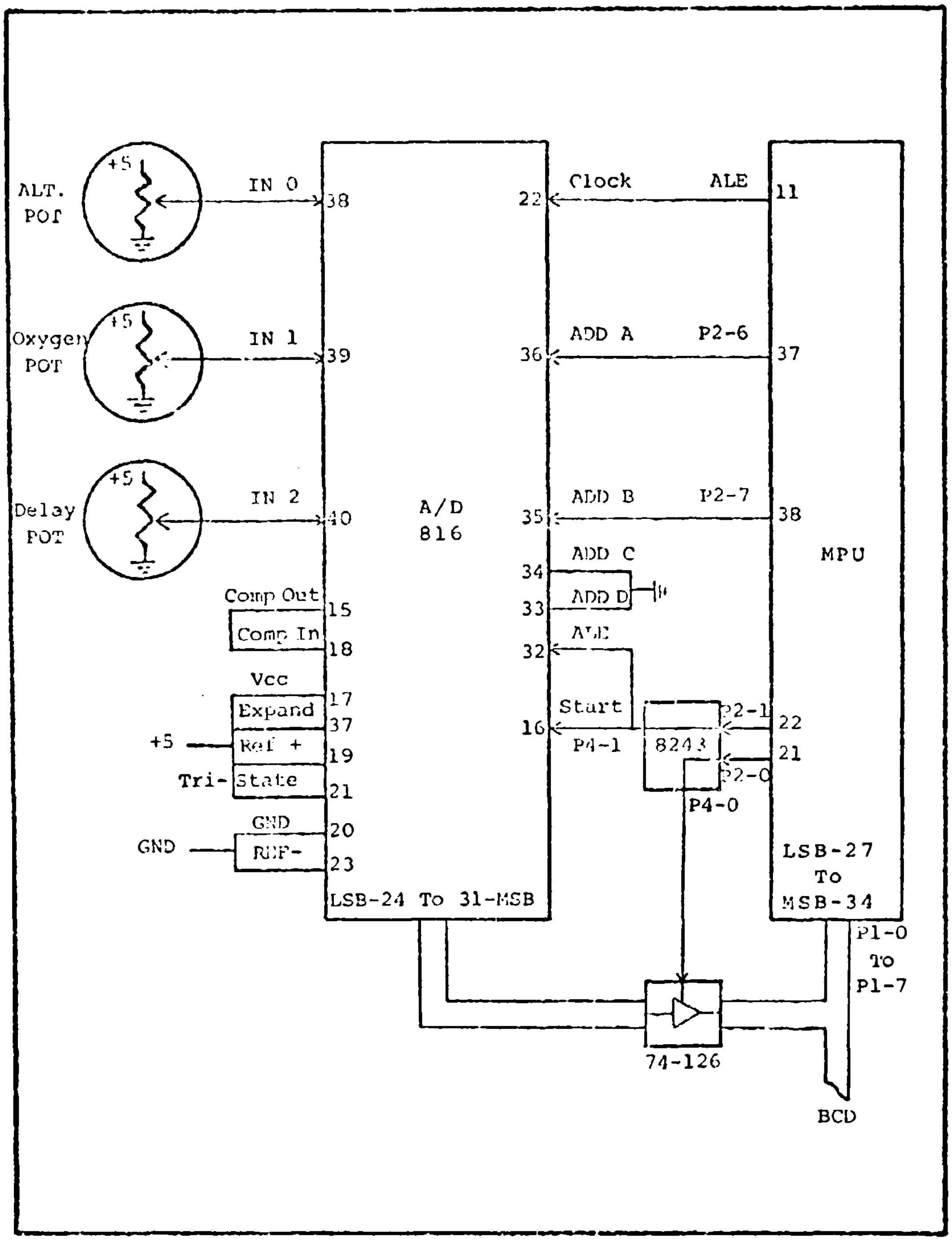


Figure 19. ADC 816 Schematics.

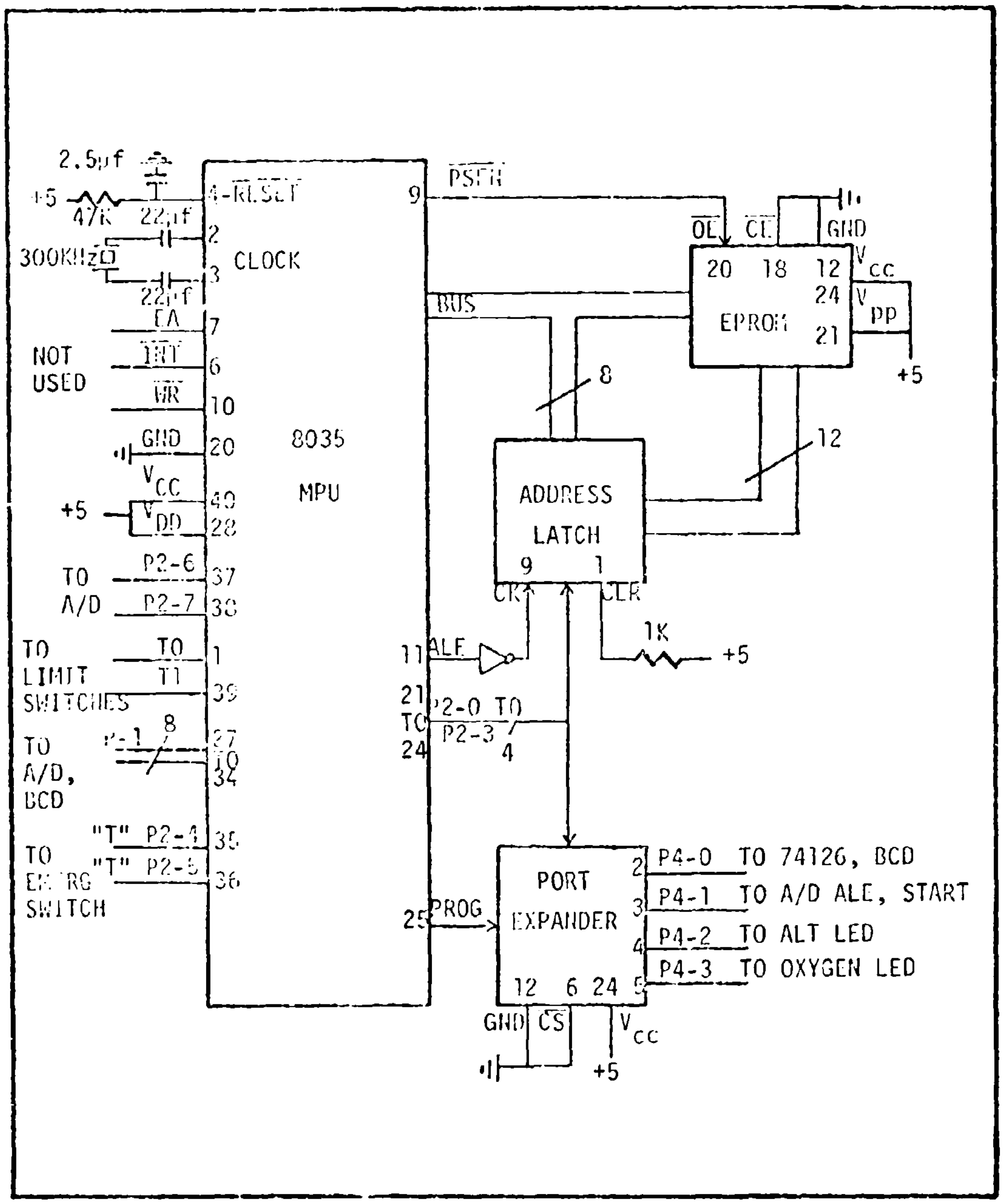
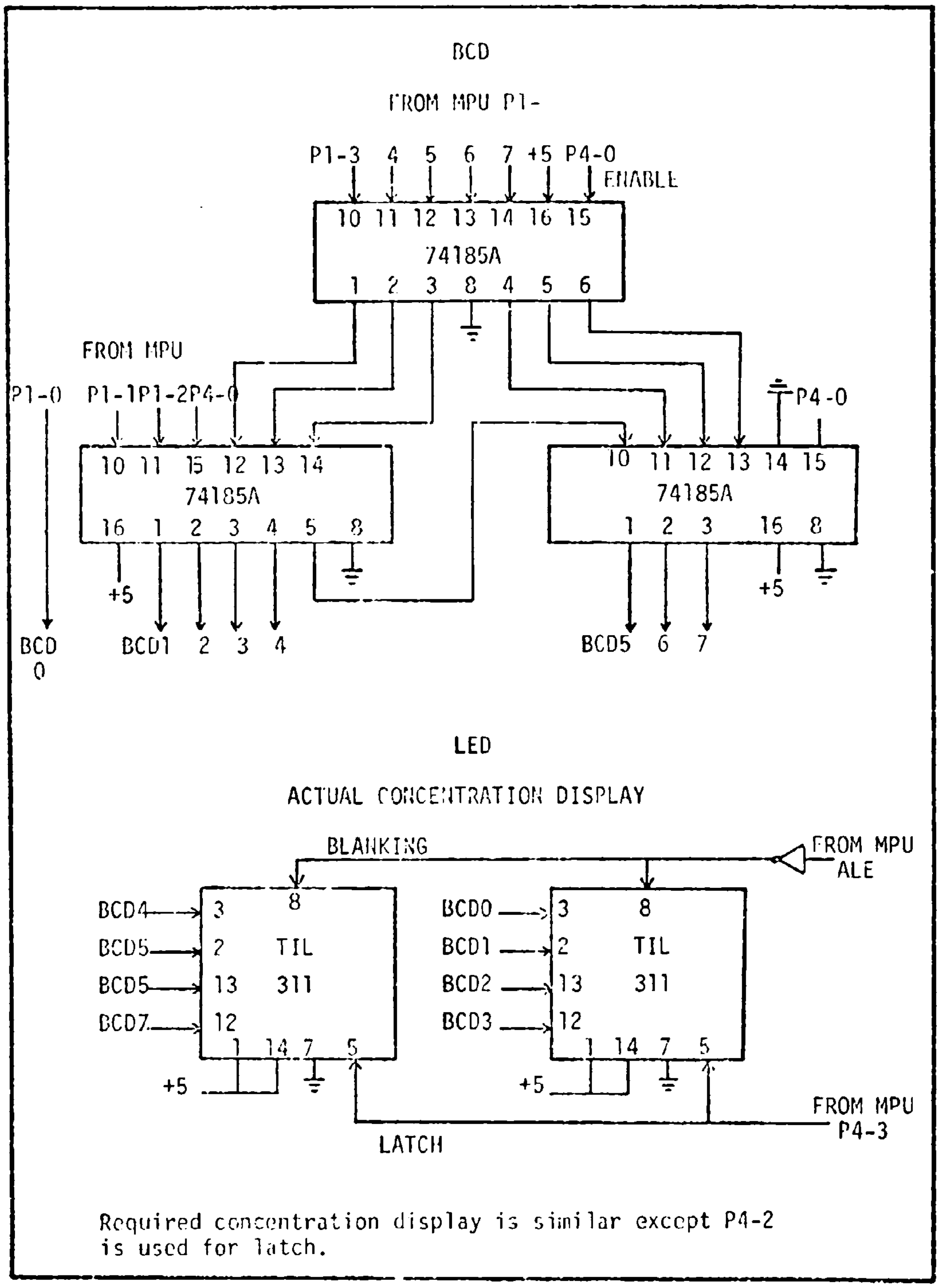


Figure 20. 8035 Schematic



Required concentration display is similar except P4-2 is used for latch.

Figure 21. BCD and LED Schematic (Ref 15:7-295)

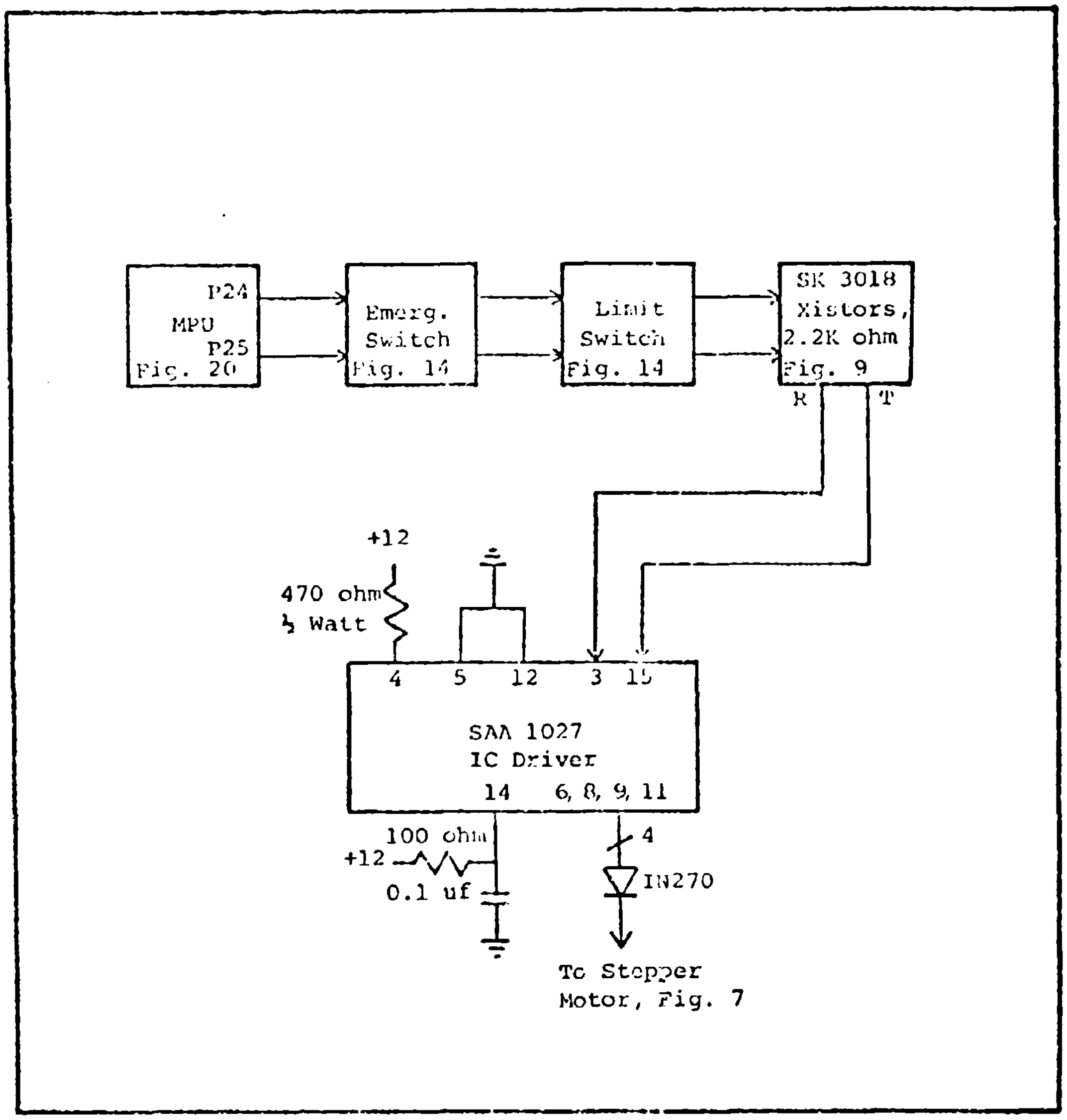


Figure 22. Motor and IC Driver Schematic.

APPENDIX B

SYSTEM SOFTWARE

APPENDIX B

SYSTEM SOFTWARE

```
ADDRESS      OPCODE      INSTRUCTION             COMMENTS

Block 1

c.  Program Port 4.
 000          23          MOV A, #                Set P4 on
 001          04          0000/0100               8243 expander
 002          3A          OUT P2, A               as an output port.

c.  Set Digital Displays.
 003          23          MOV A, #                Disable LED's so
 004          0D          000/1101                unwanted data
 005          3C          OUT P4, A               will not be displayed.

c.  Drive Motor to open position.
 006          36          JT1 to                  When open, jump to
 007          19          add. 019                next routine.
 008          23          MOV A, #                If not open, set
 009          3D          0001/1100               T high and set, R=0
                                                  to open motor.
 00A          3A          OUT P2, A               Output control signals.

    c.  Delay for Maximum Motor Speed.
 00B          B8          MOV R0, #               Use a two loop
 00C          06          06-Delay Constant       nested decrement
 00D          B9          MOV R1, #               routine with 06
 00E          FF          FF-Delay Constant       and FF selected
 00F          E9          D(R1)JNZ to             as delay constants.
 010          0F          add. 00F                Delay keeps from
 011          E8          D(R0)JNZ to             overspeeding
 012          0D          add. 00D                stepper motor.
 013          23          MOV A, #                Set T low,
 014          AD          1000/1101               select delay channel,
 015          3A          OUT P2, A               leave rotation = 0.
 016          27          CLR A                   Unconditionally,
 017          C6          JZ to                   jump to start
 018          06          add. 006                of this routine.

c.  Set Motor Position Counter
 019          BF          MOV R7, #               Set MPC
 01A          FF          255                     to 255.
 01B          00          No operation
 01C          00          NOP
 01D          00          NOP
```

c. Initialize Segmented Values, Registers 08_{16} to 17_{16}.

```
01E    B8    MOV R0, #
01F    00    00                  Value of first seg-
                                 mented register #.
020    B9    MOV R1, #
021    08    08                  First register
                                 number in hex.
022    BA    MOV R2, #
023    10    10                  Value of MPC
                                 increment step.
024    BB    MOV R3, #
025    10    10                  Register count down
                                 value in hex.
026    F8    MOV A, R0           These steps put 00_16
                                 in register
027    A1    MOV @ R1, A         number 08_16, 10_16 in
                                 register
028    6A    ADD A, R2           number 09_16, 20_16 in
                                 register
029    A8    MOV R0, A           number 0A_16...and F0_16
02A    19    INC R1              in
02B    EB    D(R3)JNZ to         register number 17_16.
02C    26    add. 026
02D    00    NOP
02E    00    NOP
02F    00    NOP
```

c. Initialize Original Register Number.

```
030    D5    SEL RB1             Load R1' with 00 to
031    B9    MOV R1', #          ensure register number
032    00    00                  changes in Test 2 on
                                 first pass
033    C5    SEL RB0             after initialization.
```

Block 2

```
c.  Read Delay Potentiometer.
  034         23         MOV A, #                     Select Delay
  035         8D         1000/1101                    Channel for
  036         3A         OUT P2, A                    A/D converter.

  037         23         MOV A, #                     Set Start and
  038         8F         1000/1111                    ALE of A/D
  039         3C         OUT P4, A                    converter high.

  03A         23         MOV A, #                     Set Start and
  03B         8D         1000/1101                    ALE of A/D
  03C         3C         OUT P4, A                    Converter low.

  03D         23         MOV A, #                     Delay for A/D
  03E         FF         FF-delay constant            successive
  03F         07         DEC A                        approximation,
  040         96         JNZ, to                      FF used for max
  041         3F         add. 03F                     motor speed.

  042         09         IN A, P1                     Read delay value.
  043         17         INC A                        Increment so delay ≠ 0.
  044         AE         MOV R6, A                    Store value in
                                                      register 6.
  045         00         NOP
  046         00         NOP
  047         00         NOP

c.  Delay Loop, Time of delay is proportional to delay pot.

  048         A9         MOV R1, A                    Move Pot value into
                                                      register 1.
  049         A8         MOV R0, A                    Move Pot value into
                                                      register 0.
  04A         FE         MOV A, R6                    Move Pot value into
                                                      accumulator.
  04B         07         DEC A                        Decrement accumulator
                                                      until it
  04C         96         JNZ, to                      equals zero.
  04D         4B         add. 04B                     First Loop.
  04E         E8         D(R0)JNZ, to                 Decrement R0 until
  04F         4A         add. 04A                     it equals zero. Second
                                                      Loop.
  050         E9         D(R1)JNZ, to                 Decrement R1 until
                                                      it equals
  051         4A         add. 04A                     zero.  Third Loop.
  052         00         NOP
  053         00         NOP
  054         00         NOP

c.  Test Limit Switches and Reset MPC if Appropriate.
  055         46         JNT1 to                      Test for closed.
                                                      If T1 equals zero,
056        59        add. 059              jump to open test.

057        BF        MOV R7, #             If closed, set
058        00        00                    MPC to 00.
059        26        JNTO to               Test for open.  If
                                           TO equals zero,
05A        5D        add. 05D              jump to end of this
                                           routine.

05B        BF        MOV R7, #             If open, set
05C        FF        FF=255_10             MPC to 255_10.
05D        00        NOP
05E        00        NOP
05F        00        NOP
```

c. Read Altitude Sensor, Look-up Required Oxygen Concentration, Output Value to Displays.

```
060        23        MOV A, #              Select altitude
061        0D        0000/1101             Channel on
062        3A        OUT P2, A             A/D converter.

063        23        MOV A, #              Set Start and
064        0F        0000/1111             ALE on A/D
065        3C        OUT P4, A             high.

066        23        MOV A, #              Set Start
067        0D        0000/1111             and ALE on
068        3C        OUT P4, A             A/D low.

069        23        MOV A, #              Delay
06A        20        20-delay constant     for
06B        07        DEC A                 A/D
06C        96        JNZ, to               successive
06D        6B        add. 06B              approximation.

06E        09        IN A, P1              Read Altitude Sensor,
06F        A8        MOV R0, A             store in register 0.

070        F8        MOV A, R0             Jump to page 2
071        00        NOP                   to look-up
072        44        JMP, Page 2           altitude schedule
073        FA        add. 2FA              and return
074        00        NOP                   to address 074.

075        23        MOV A, #              Set LED
076        08        0000/1000             latch to accept
077        3C        OUT P4, A             a new value.

078        FC        MOV A, R4             Output required oxygen
079        39        OUTL P1, A            schedule to LED's.

07A        23        MOV A, #              Set LED latch to
07B        0D        0000/1101             lock-in
07C        3C        OUT P4, A             above value.
```

c. Read Oxygen Sensor, Look-up Actual Oxygen Concentration, Output Value to Displays.

```
07D        23        MOV A, #              Select oxygen
07E        4D        0100/1101             Channel on
07F        3A        OUT P2, A             A/D converter.

080        23        MOV A, #              Set Start and
081        4F        0100/1111             ALE on A/D
082        3C        OUT P4, A             high.

083        23        MOV A, #              Set Start
084        4D        0100/1101             and ALE
085        3C        OUT P4, A             on A/D low.

086        23        MOV A, #              Delay
087        20        20-delay constant     for
088        07        DEC A                 A/D
089        96        NJZ, to               successive
08A        88        add. 088              Approximation.

08B        09        IN A, P1              Read oxygen sensor,
08C        A9        MOV R1, A             store in register 1.

08D        F9        MOV A, R1             Jump to page 3
08E        00        NOP                   to look-up
08F        64        JMP, Page 3           oxygen value
090        FA        add. 0FA              and return
091        00        NOP                   to address 091.

092        23        MOV A, #              Set LED latch
093        44        0100/0100             to accept
094        3C        OUT P4, A             a new value.

095        FD        MOV A, R5             Output actual oxygen
096        39        OUT P1, A             schedule to LED's.

097        23        MOV A, #              Set LED latch
098        0D        0000/1101             to lock-in
099        3C        OUT P4, A             above value.
09A        00        NOP
 .          .        .
 .          .        .
 .          .        .
09F        00        NOP
```

```
c.  Set Previous Segmented Register Number of Operation in R0'
c.  and Determine new Register Number of Corresponding
c.  Oxygen Segment.
c.  R1 = R1' = new reg. no.  R0' = last reg. no.  R4 = required oxygen
c.  concentration

 0A0        FC        MOV A, R4              Put required oxygen
 0A1        D5        SEL RB1                concentration in
 0A2        AC        MOV R4', A             register R4'.

 0A3        F9        MOV A, R1'             Put previous register
 0A4        A8        MOV R0', A             number in R0'.

 0A5        B9        MOV R1', #             Set R1' to 17_16 which
 0A6        17        17                     is the reg. no. of the
                                             bottom segment.
 0A7        BE        MOV R6', #             Set R6' to 26% which is
 0A8        1A        1A_16=26_10%           the lower value of the
                                             second segment.
 0A9        97        CLR C                  Clear the carry bit
 0AA        FE        MOV A, R6'             and subtract 26%
 0AB        37        CPL A                  from the required
 0AC        17        INC A                  OXYGEN
 0AD        6C        add. A, R4'            concentration.

 0AE        E6        JNC to                 If carry equals 0, jump
 0AF        B8        add. 0B8               to end of this routine,
                                             otherwise
 0B0        C9        DEC R1'                decrement new reg. no.,
 0B1        23        MOV A, #               and add 05_16 which
 0B2        05        05_16=step value       is the step value
 0B3        6E        ADD  A, R6'            to register R6' and
 0B4        AE        MOV R6', A             place in R6'.

 0B5        27        CLR A                  Unconditionally jump
 0B6        C6        JZ, to                 to clear
 0B7        A9        add. 0A9               carry instruction.

 0B8        F9        MOV A, R1'             Place new
 0B9        C5        SEL RB0                register no.
 0BA        A9        MOV R1, A              in register R1.
 0BB        00        NOP
  .          .        .
  .          .        .
  .          .        .
 0BF        00        NOP

c.  Calculate and Output Rotation

 0C0        97        CLR C                  Clear carry for future
                                             test.
 0C1        FD        MOV A, R5              If R5, which is the actual
 0C2        C6        JZ to                  concentration, equals
                                             zero, jump
 0C3        E0        add. 0E0               to address 0E0.
```

```
0C4    37    CPL A              Subtract actual oxygen
0C5    17    INC A              concentration from required
0C6    6C    ADD  A, R4         concentration and store
0C7    AA    MOV R2, A          results in R2 for future use

0C8    FA    MOV A, R2          If oxygen error, results
0C9    C6    JZ to              from above, equals zero,
0CA    DB    add. 0DB           jump to address 0DB.

0CB    F6    JC to              If carry is set, jump
0CC    DB    add. 0DB           to address 0DB, else

0CD    FA    MOV A, R2          invert the magnitude
0CE    37    CPL A              or oxygen error, since
0CF    17    INC A              error is negative if
0D0    AA    MOV R2, A          this routine is entered.

0D1    BB    MOV R3, #          Set rotation to 1 to
0D2    20    0010/0000          decrease oxygen concentra-
                                tion.
0D3    27    CLR A              Unconditionally
0D4    C6    JZ to              jump to
0D5    E4    add. 0E4           address 0E4.
0D6    00    NOP
 .      .     .
 .      .     .
 .      .     .
0DA    00    NOP
0DB    BB    MOV R3, #          Set rotation from
0DC    00    0000/0000          carry test to increase
                                direction.
0DD    27    CLR A              Unconditionally,
0DE    C6    JZ to              jump to
0DF    E4    add. 0E4           address 0E4.

0E0    BB    MOV R3, #          Entry here is if actual
0E1    00    0000/0000          concentration equals zero,
                                so increase rotation,
0E2    FC    MOV A, R4          and set oxygen error equal
0E3    AA    MOV R2, A          to required concentration.

0E4    23    MOV A, #           Combine rotation direction
0E5    0D    0000/1101          with LED control and
0E6    4B    ORL A, R3          output results
0E7    AB    MOV R3, A          to the motor
0E8    3A    OUT P2, A          IC driver and LED's.
0E9    00    NOP
 .      .     .
 .      .     .
 .      .     .
0EF    00    NOP
```

Test 1

c. Determine if required number of motor increments is less than 6%.

```
0F0           97          CLR C                     Clear carry bit for future
                                                    test.
0F1           23          MOV A, #                  Subtract 6
0F2           06          06₁₆                      from the
0F3           37          CPL A                     oxygen
0F4           17          INC A                     error
0F5           6A          ADD  A, R2                and
0F6           E6          JNC, to                   if the carry is set,
0F7           FA          add. 0FA                  jump to address 0FA.
0F8           24          JMP, page 1,              Jump to
0F9           26          add. 126                  address 126.

0FA           24          JMP, page 1               Jump to
0FB           00          add. 100                  address 100.
0FC           00          NOP
 .             .           .
 .             .           .
 .             .           .
0FF           00          NOP
```

Block 3

c. Calculate and Output T, set MPC.

```
100     FA      MOV A, R2         If the oxygen error
101     C6      JZ to             equals zero,
102     12      add. 112          jump to address 112, else

103     23      MOV A, #          combine rotation and
104     1D      0001/1101         LED control with signal to
105     4B      ORL A, R3         increment stepper motor
106     3A      OUT P2, A         and output signals.

107     FB      MOV A, R3         Set MPC in rotation direction.
108     B2      J Bit 5 to        If rotation bit is high,
109     0E      add. 10E          jump to address 10E, else
10A     CF      DEC R7            decrement motor position
                                  counter.
10B     27      CLR A             Unconditionally
10C     C6      JZ to             jump to
10D     14      add. 114          address 114.
10E     1F      INC R7            Increment motor position
                                  counter.
10F     27      CLR A             Unconditionally
110     C6      JZ to             jump to
111     14      add. 114          address 114.

112     FB      MOV A, R3         Oxygen error is zero, so do
113     3A      OUT P2, A         not increment motor.
114     00      NOP
115     00      NOP
 .       .       .
 .       .       .
 .       .       .
119     00      NOP
```

c. Determine if Update is valid and Update Segment Value.

```
11A     FA      MOV A, R2         If oxygen error is
11B     96      JNZ to            not zero, jump to
11C     1F      add. 11F          address 11F, else update
11D     FF      MOV A, R7         by placing motor position
11E     A1      MOV @ R1, A       counter in segment register
11F     00      NOP               of current segment.
120     04      JMP Page 0,       Unconditionally jump to
121     34      add. 034          address 034, block 2.
122     00      NOP
 .       .       .
 .       .       .
 .       .       .
125     00      NOP
```

Test 2

c. Determine if Register Number has changed.

```
126     D5     SEL RB 1        Subtract present register
127     F8     MOV A, R0       number (segment of
128     37     CPL A           operation) from previous
129     17     INC A           register number.
12A     69     ADD  A, R1'     If result of
12B     C5     SEL RB0         subtraction is
12C     C6     JZ to           zero, jump to
12D     5A     add. 15A        address 15A, else continue.
12E     00     NOP
12F     00     NOP
```

Block 4

```
c.  Drive Motor to Segment Position at Maximum Rate
c.  while adjusting the Motor Position Counter.

 130          F1          MOV A @ R1              If the segment
 131          37          CPL A                   value, R1, equals
 132          17          INC A                   the motor positon
 133          6F          ADD  A, R7              counter, R7, then
 134          C6          JZ to                   jump to address
 135          55          add. 155                155, else continue.

 136          23          MOV A, #                Set T high and
 137          1D          0001/1101               combine with LED
 138          4B          ORL A, R3               and R signals and
 139          3A          OUT P2, A               output results.

 13A          FB          MOV A, R3               addresses 13A to 142
 13B          B2          J Bit 5 to              are the same as
 13C          41          add. 141                addresses 107 to 10E.
 13D          CF          DEC R7                  The motor position
 13E          27          CLR A                   counter is
 13F          C6          JZ to                   incremented or
 140          42          add. 142                decremented
 141          1F          INC R7                  according to
 142          00          NOP                     the rotation.

 143          FB          MOV A, R3               Set T low for
 144          3A          OUT P2, A               future incrementing.
  .            .          .
  .            .          .
  .            .          .
 147          00          NOP

c.  Delay for maximum Motor Speed.

 148          D5          SEL RB1                 Use a double
 149          BA          MOV R2', #              nested loop
 14A          08          08=delay constant       to create a
 14B          BB          MOV R3', #              software delay.
 14C          FF          FF=delay constant       Constants of 08,
 14D          EB          D(R3')JNZ, to           and FF are
 14E          4D          add. 14D                decremented until
 14F          EA          D(R2')JNZ, to           they equal zero.
 150          4B          add. 14B                This keeps from
 151          C5          SEL RB0                 overspeeding motor.

 152          27          CLR A                   Unconditionally
 153          C6          JZ, to                  jump to
 154          30          add. 130                address 130.

 155          04          JMP, Page 0,            Unconditionally jump
 156          34          add. 034                to address 034, Block 2.
 157          00          NOP
 158          00          NOP
 159          00          NOP
```

Block 5

c. Increment Motor , MPC and Unconditionally Update

```
15A     23     MOV A, #          Set T high and combine
15B     1D     0001/1101         with LED and Rotation
15C     4B     ORL A, R3         signals.
15D     3A     OUT P2, A         Output results.

15E     FB     MOV A, R3         Address 15E to
15F     B2     J Bit 5, to       166 are the same as 107 to
160     65     add. 166          10E.  The motor position
161     CF     DEC R7            counter is incremented
162     27     CLR A             or decremented
163     C6     JZ to             according to the
164     66     add. 166          direction of
165     1F     INC R7            rotation as previously
166     00     NOP               determined.

167     FB     MOV A, R3         Set T low for
168     3A     OUT P2, A         future increments.

169     FF     MOV A, R7         Unconditionally update
16A     A1     MOV @ R1, A       segmented value to MPC.

16B     04     JMP Page 0,       Unconditionally jump
16C     34     add. 034          to address 034, Block 2.

16D     00     NOP
 .       .      .
 .       .      .
 .       .      .
1FF     00     NOP
```

c. Altitude Schedule Table

```
200      14      Load values which correspond
201      15      to required oxygen
 .        .      concentration.
 .        .
 .        .
2F9      64

2FA      A3      MOVP A @ A          Use value of altitude sensor
2FB      AC      MOV R4, A           output as address for
2FC      04      JMP, Page 0,        table look-up.  Return to
2FD      74      add. 074            main program with required
2FE      00      NOP                 oxygen concentration
2FF      00      NOP                 stored in register 4.
```

c. Oxygen Look-up Table

```
300      00      Load values which correspond
301      01      to actual oxygen
 .        .      concentration.
 .        .
 .        .
3F9      64

3FA      A3      MOV A @ A           Use value of oxygen
3FB      AD      MOV R5, A           sensor output as an
3FC      04      JMP, Page 0         address for table look-up.
3FD      91      add. 091            Return to main program
3FE      00      NOP                 with actual oxygen
                                     concentration
3FF      00      NOP                 stored in register 5.
```

Vita

Thomas C. Horch, born 11 April 1949, graduated from high school in Austin, Texas in 1967. He then attended the University of Texas at Austin and graduated with a Bachelor of Science degree in Electrical Engineering in 1971. After graduation, he was commissioned in the United States Air Force through Officer Training School. Next, he attended Undergraduate Navigator Training School and Electronic Warfare Officer Training at Mather AFB, California. He then attended Basic Survival School at Fairchild AFB, Washington; Water Survival School at Homestead AFB, Florida; and Jungle Survival School at Clark AFB, Philippines. He then served in Thailand where he was an instructor and flight examiner as an Electronic Warfare Officer in the AC-130 gunships. In 1975 he was assigned to the DC-130 program at Davis-Monthan AFB, Arizona. While stationed there, he worked as a U-2 navigator, and he later became an instructor and flight examiner for flying remotely piloted vehicles. He has attended SOS in residence and entered the School of Engineering, Air Force Institute of Technology in June 1979.

Permanent address: 2217 Onion Creek Parkway #109
Austin, Texas 78747

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GEO/EE/80D-2	2. GOVT ACCESSION NO. AD-A100 823	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DIGITAL CONTROL SYSTEM FOR AN ON-BOARD OXYGEN GENERATOR		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) THOMAS C. HORCH CAPT USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB OH		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS School of Aerospace Medicine Brooks AFB TX		12. REPORT DATE December 1980
		13. NUMBER OF PAGES 85
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release IAW AFR 190-17

[illegible]

16 JUN 1981

FREDERICK C. LYNCH, Major, USAF
Director of Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

On-Board Oxygen Generator	Analog-to-Digital Converter
Microprocessor	Active Control
Stepper Motor	Digital Control System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A prototype on-board oxygen generation system (OBOGS) is being tested by the USAF School of Aerospace Medicine (SAM). The OBOGS is a candidate to replace present liquid oxygen systems for aircrew consumption on manned aircraft. The OBOGS passes outside air through a molecular sieve to produce an oxygen enriched breathing product. Oxygen concentration of the OBOGS's output is controlled by a purge orifice valve. The SAM envisions using a digital system to control the OBOGS. →

(Continued on Reverse)

BLOCK 20 - Abstract (Cont'd):

A digital control system for the OBOGS was developed and consists of a stepper motor, microprocessor, system sensors, support circuitry, and software. The control system software is a collection of instructions which allow the MPU to read data from sensors, to interpret that data, and to issue system hardware control signals. System software was fairly complex as methods were employed to compensate for the OBOGS's lengthy response time. This was accomplished by using a segmented table. If motor drive is anticipated to be time-consuming, a software routine is used to preposition the motor to a predetermined location within the segmented table. This position is updated when more accurate information is available.

A prototype system was constructed and tested in the laboratory. The control system successfully controlled the stepper motor.